แอดวานซ์ซิงเกิลบอร์ด

Advanced Single Board

โดย

นายพีรชาติ จริยสุทธิศักดิ์ เลขประจำตัว 38014353

นายภาณุพงษ์ รัตน์พงษ์โสภิต เลขประจำตัว 38014372

นายมนตรี ศรีสำราญ เลขประจำตัว 38014389

อาจารย์ที่ปรึกษา

ผศ.ดร.สุรพันธุ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ ปีการศึกษา 2541

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง แอดวานซ์ซิงเกิลบอร์ด

ผู้จัดทำ

1. นายพีรชาติ จริยสุทธิศักดิ์
2. นายภาณุพงษ์ รัตน์พงษ์โสภิต
3. นายมนตรี ศรีสำราญ

..... อาจารย์ที่ปรึกษา

(ผศ.ดร.สุรพันธุ์ เอื้อไพบูลย์)

แอดวานซ์ซิงเกิลบอร์ด

Advanced Single Board

นายพีรชาติ จริยสุทธิศักดิ์ เลขประจำตัว 38014353

นายภาณุพงษ์ รัตน์พงษ์โสภิต เลขประจำตัว 38014372

นายมนตรี ศรีสำราญ เลขประจำตัว 38014389

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์)

อาจารย์ที่ปรึกษา

แอดวานซ์ซิงเกิลบอร์ด

นาย พีรชาติ จริยสุทธิศักดิ์
นาย ภาณุพงษ์ รัตน์พงษ์โสภิต
นาย มนตรี ศรีสำราญ
ผศ.ดร.สุรพันธุ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ในโครงงานนี้เป็นการนำเอาไมโครคอนโทรลเลอร์ ตระกูล MCS-51 เบอร์80C31มาประยุกต์ใช้งานเป็นซิงเกิลบอร์ด(SingleBoard)ซึ่งจะประกอบด้วยส่วนที่เป็นไมโครคอนโทรลเลอร์, ส่วนของหน่วยความจำ (Memory), วงจรถอดรหัสแอดเดรสของอุปกรณ์ทั้งหมด (I/O Decoder), จอแสดงผล(LCD),คีย์บอร์ด(Keyboard),การสื่อสารผ่านพอร์ทอนุกรม(Serial Port) และวงจรวอทช์ดอก (Watch Dog) โดยลักษณะพิเศษของบอร์ดนี้คือสามารถเขียนโปรแกรมได้บนซิงเกิลบอร์ดเอง โดยไม่ต้องใช้ไมโครคอมพิวเตอร์ เพราะ มีโปรแกรมมอนิเตอร์ (Program Monitor), โปรแกรมอีดิเตอร์ (Program Editor),โปรแกรมแอสเซมเบลอร์(Program Assembler) อยู่ภายในบอร์ด จึงทำให้มีความสะดวกอย่างมากในการนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง อาทิเช่น สามารถใช้อินพุท/เอาท์พุทพอร์ทในการควบคุมสิ่งต่าง ๆ โดยไม่ต้องทำการต่อวงจรของไมโครคอนโทรลเลอร์เองเพียงแต่นำพอร์ทไปใช้งาน, สามารถเขียนโปรแกรมทดสอบบนบอร์ดเป็นภาษาแอสเซมบลีได้เลย และสามารถบันทึกไฟล์กลับไปเก็บบนไมโครคอมพิวเตอร์ได้ทั้งที่เป็นไฟล์ภาษาแอสเซมบลีและไฟล์ที่เป็นอินเทลเฮกซ์ไฟล์ (Intel hex file)

Advanced Single Board

Peerachart Jariyasutthisak
Panupong Ratpongsopit
Montri Srisamrarn
Surapan Airpaiboon Advisor
1998

Abstract

This project uses microcontroller 80C31 which is in MCS-51 series to build single board which is composed of microcontroller, memory, I/O decoder, liquid crystal display (LCD), keyboard, serial port and watch dog. The special characteristics of this board is writing program on single board without using microcomputer because it has program monitor, program editor and program assembler . It is convenient to apply in microcontroller application. For example, you can use input/output port to control anything. You don't make circuit of microcontroller, you only take the port to use. You can write program to test on board in assembly program and you can save file to microcomputer both assembly file and intel hex file.

สารบัญ

	หน้า
บทคัดย่อ	I
Abstract	II
สารบัญ	III
สารบัญรูปภาพ	VI
สารบัญตาราง	X
บทที่ 1 บทนำ	1
1.1 จุดประสงค์ของการจัดทำโครงงาน	1
1.2 ประโยชน์และการประยุกต์ใช้งาน	2
บทที่ 2 ไมโครคอนโทรลเลอร์ MCS-51	3
2.1 คุณสมบัติของไมโครคอนโทรลเลอร์ MCS-51	3
2.2 โครงสร้างภายใน MCS-51	5
2.3 รีจิสเตอร์ของ 8031	11
2.4 หน่วยประมวลผลทางคณิตศาสตร์และลอจิก	17
2.5 หน่วยความจำของ 8031	18
2.6 เวลาของการประมวลผลชุดคำสั่ง	19
2.7 การทำงานของ 8031	20
2.8 การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ 8031	23
2.9 คำสั่งการใช้งานพอร์ทอินพุท/เอาท์พุท	25
2.10 ตัวอย่างวงจรการพ่วงต่อกับอุปกรณ์ภายนอก	27
บทที่ 3 การออกแบบโครงสร้างทางฮาร์ดแวร์	29
3.1 วงจรควบคุมหลัก	29
3.2 การถอดรหัสตำแหน่งของหน่วยความจำทั่วไป	31
3.3 การถอดรหัสตำแหน่งของ I/O	33
3.4 หน่วยแสดงผล	34
3.5 คีย์บอร์ด	35
3.6 การติดต่อกับพอร์ทพริมเตอร์	36
3.7 วงจรสื่อสารผ่านพอร์ทอนุกรม	37
3.8 วงจรสร้างฐานเวลาจริง	37

3.7 วงจรสื่อสารผ่านพอร์ทอนุกรม 37
3.8 วงจรสร้างฐานเวลาจริง 37
3.9 วงจรวอทช์ดอก 38
3.10 ลำโพง 40
3.11 วงจรจ่ายไฟเลี้ยง 40
บทที่ 4 การออกแบบโครงสร้างทางด้านซอร์ฟแวร์ 41
4.1 เมนูย่อยของคำสั่งเกี่ยวกับโปรแกรม 44
4.2 เมนูย่อยของคำสั่งเกี่ยวกับหน่วยความจำและรีจิสเตอร์ 46
4.3 เมนูย่อยของคำสั่งเกี่ยวกับการสื่อสารทางพอร์ทอนุกรม 59
4.4 เมนูย่อยของคำสั่งอินพุทและเอาท์พุท 65
4.5 เมนูย่อยของคำสั่งอื่นๆ 67
4.6 การออกจากโปรแกรม 69
บทที่ 5 ผลการทดลอง 71
5.1 จอภาพเมื่อเริ่มการใช้งาน 71
5.2 การใช้ โปรแกรมแอสแซมเบลอร์ 72
5.3 การติดต่อกับหน่วยความจำและรีจิสเตอร์ 75
5.4 การติดต่อกับไมโครคอมพิวเตอร์ 80
5.5 การส่งข้อมูลออกทางพอร์ทของผู้ใช้ (USER PORT) 82
5.6 เครื่องมืออื่น ๆ 83
บทที่ 6 สรุปและวิจารณ์โครงงาน 85
6.1 สรุปผลการปฏิบัติงาน 85
6.2 ปัญหาและอุปสรรคในการทำงาน 85
6.3 แนวทางการแก้ไข 86
6.4 สรุป 87
ภาคผนวก 89
ภาคผนวก ก ชุดคำสั่งภายใน MCS-51
ภาคผนวก ข การใช้งาน 8255
ภาคผนวก ค การใช้งาน LCD
ภาคผนวก ง การใช้งาน MAX 232
ภาคผนวก จ โปรแกรมมอนิเตอร์

กิตติกรรมประกาศ
เอกสารอ้างอิง

สารบัญรูปภาพ

รูปที่ 2.1 บล็อกไดอะแกรมของ MCS-51 4

รูปที่ 2.2 ตำแหน่งของรีจิสเตอร์ต่าง ๆ และหน่วยความจำภายใน 5

รูปที่ 2.3 การจัดวางขาของ 8031 6

รูปที่ 2.4 โครงสร้างของพอร์ท 0 6

รูปที่ 2.5 โครงสร้างของพอร์ท 1 7

รูปที่ 2.6 โครงสร้างของพอร์ท 2 7

รูปที่ 2.7 โครงสร้างของพอร์ท 3 8

รูปที่ 2.8 ค่ารีจิสเตอร์เมื่อเกิดการรีเซ็ท 8031 9

รูปที่ 2.9 วงจรออสซิลเลเตอร์ภายใน 8031 10

รูปที่ 2.10 8031 ที่ทำงานโดยสัญญาณนาฬิกาที่มาจากภายนอก 10

รูปที่ 2.11 ตำแหน่งของรีจิสเตอร์หน้าที่พิเศษต่าง ๆ 11

รูปที่ 2.12 แสดงการทำงานของรีจิสเตอร์โปรแกรมเคาน์เตอร์ 13

รูปที่ 2.13 บิตต่าง ๆ ภายในรีจิสเตอร์ PSW 15

รูปที่ 2.14 บิตต่าง ๆ ภายในรีจิสเตอร์ PCON 16

รูปที่ 2.15 แผนภาพแสดงหน่วยประมวลผลทางคณิตศาสตร์ 17

รูปที่ 2.16 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับ 8031 18

รูปที่ 2.17 ผังหน่วยความจำสำหรับเก็บข้อมูลของ 8031 18

รูปที่ 2.18 ผังการทำงานของแต่ละคำสั่งใน 8031 22

รูปที่ 2.19 ลำดับการทำงานแบบโพลลิ่ง 24

รูปที่ 2.20 แผนภาพการทำงานอย่างง่ายของการติดต่อระหว่างอุปกรณ์ภายนอก กับหน่วยความจำ 25

รูปที่ 2.21 วงจรไมโครคอนโทรลเลอร์ที่ใช้อีพรอมภายนอก 26

รูปที่ 3.1 ส่วนประกอบของซิงเกิลบอร์ด 28

รูปที่ 3.2 วงจรรวมของซิงเกิลบอร์ด 29

รูปที่ 3.3 วงจรถอดรหัสหน่วยความจำ 31

รูปที่ 3.4 วงจรถอดรหัสตำแหน่งของ I/O 32

รูปที่ 3.5 วงจรของหน่วยแสดงผล 33

รูปที่ 3.6 วงจรสแกนคีย์บอร์ด 34

รูปที่ 3.7 การติดต่อกับพอร์ทพรินเตอร์ 34
รูปที่ 3.8 วงจรสื่อสารผ่านพอร์ทอนุกรม 35
รูปที่ 3.9 วงจรสร้างฐานเวลาจริง 36
รูปที่ 3.10 วงจรวอทช์ดอก 37
รูปที่ 3.11 วงจรขับลำโพง 38
รูปที่ 3.12 วงจรจ่ายไฟเลี้ยง 39
รูปที่ 4.1 โครงสร้างของเมนูหลัก 41
รูปที่ 4.2 โฟลว์ชาร์ทแสดงการทำงานของเมนูหลัก 42
รูปที่ 4.3 โครงสร้างของเมนูย่อยในส่วนของโปรแกรม 43
รูปที่ 4.4 โฟลว์ชาร์ทการทำงานของเมนูย่อยโปรแกรม 43
รูปที่ 4.5 โฟลว์ชาร์ทการทำงานของโปรแกรมแอสแซมเบลอร์ และ อันแอสแซมเบลอร์ 45
รูปที่ 4.6 รูปแบบของออปโคด 46
รูปที่ 4.7 โฟลว์ชาร์ทแสดงการทำงานของการรันโปรแกรม 47
รูปที่ 4.8 การจัดสรรหน่วยความจำของซิงเกิลบอร์ด 47
รูปที่ 4.9 โฟลว์ชาร์ทการทำงานของเมนูย่อยเกี่ยวกับหน่วยความจำ 48
รูปที่ 4.10 โฟลว์ชาร์ทแสดงการอ่านข้อมูลของหน่วยความจำโปรแกรม 49
รูปที่ 4.11 เมนูย่อยของคำสั่งเกี่ยวกับหน่วยความจำข้อมูล 49
รูปที่ 4.12 โฟลว์ชาร์ทการทำงานของเมนูย่อยเกี่ยวกับหน่วยความจำข้อมูล 50
รูปที่ 4.13 โฟลว์ชาร์ทแสดงการอ่านข้อมูลจากหน่วยความจำข้อมูล 51
รูปที่ 4.14 โฟลว์ชาร์ทแสดงการเขียนข้อมูลใส่หน่วยความจำข้อมูล 51
รูปที่ 4.15 โฟลว์ชาร์ทแสดงการคัดลอกข้อมูล 52
รูปที่ 4.16 โฟลว์ชาร์ทแสดงการเติมข้อมูลเป็นช่วง 53
รูปที่ 4.17 โฟลว์ชาร์ทแสดงการค้นหาข้อมูล 54
รูปที่ 4.18 เมนูย่อยของคำสั่งเกี่ยวกับหน่วยความจำข้อมูลภายใน 55
รูปที่ 4.19 โฟลว์ชาร์ทแสดงการทำงานของคำสั่งเกี่ยวกับหน่วยความจำข้อมูลภายใน 55
รูปที่ 4.20 โฟลว์ชาร์ทแสดงการอ่านข้อมูลบนหน่วยคามจำข้อมูลภายใน 56
รูปที่ 4.21 โฟลว์ชาร์ทแสดงการเขียนข้อมูลบนหน่วยความจำข้อมูลภายใน 56
รูปที่ 4.22 โฟลว์ชาร์ทแสดงการหาข้อมูลในหน่วยความจำข้อมูลภายใน 57
รูปที่ 4.23 เมนูย่อยของคำสั่งเกี่ยวกับรีจิสเตอร์ฟังก์ชั่นพิเศษ 57
รูปที่ 4.24 โฟลว์ชาร์ทแสดงการทำงานของคำสั่งเกี่ยวกับรีจิสเตอร์ฟังก์ชั่นพิเศษ 58
รูปที่ 4.25 โฟลว์ชาร์ทแสดงการอ่านข้อมูลบนรีจิสเตอร์ฟังก์ชั่นพิเศษ 58

รูปที่ 4.26 โฟลว์ชาร์ทแสดงการเขียนข้อมูลบนรีจิสเตอร์ฟังก์ชั่นพิเศษ 59
รูปที่ 4.27 เมนูย่อยของคำสั่งเกี่ยวกับการสื่อสารทางพอร์ทอนุกรม 59
รูปที่ 4.28 โฟลว์ชาร์ทแสดงการทำงานของคำสั่งเกี่ยวกับการสื่อสารทางพอร์ทอนุกรม 60
รูปที่ 4.29 โฟลว์ชาร์ทแสดงการคัดลอกโปรแกรมจากพีซี 61
รูปที่ 4.30 โฟลว์ชาร์ทแสดงการคัดลอกข้อมูลจากบอร์ดสู่พีซีแบบ Intel Hex File 62
รูปที่ 4.31 โฟลว์ชาร์ทแสดงการคัดลอกข้อมูลจากบอร์ดสู่พีซีเป็นโปรแกรมแอสแซมบลี 63
รูปที่ 4.32 การตั้งค่าความเร็วของการติดต่อทางพอร์ทอนุกรม 64
รูปที่ 4.33 เมนูย่อยของคำสั่งอินพุทและเอาท์พุททางพอร์ท 8255 65
รูปที่ 4.34 แสดงการทำงานของคำสั่งอินพุทและเอาท์พุททางพอร์ท 8255 65
รูปที่ 4.35 โฟลว์ชาร์ทการอินพุทค่าทางพอร์ท 8255 66
รูปที่ 4.36 โฟลว์ชาร์ทการเอาท์พุทค่าทางพอร์ท 8255 66
รูปที่ 4.37 เมนูย่อยของคำสั่งอื่นๆ 67
รูปที่ 4.38 โฟลว์ชาร์ทการทำงานของคำสั่งอื่นๆ 67
รูปที่ 4.39 โฟลว์ชาร์ทการทำงานของวงจรเช็คการเชื่อมต่อ 68
รูปที่ 4.40 โฟลว์ชาร์ทแสดงการเล่นดนตรี 69
รูปที่ 4.41 โฟลว์ชาร์ทแสดงการป้องกันการใช้งานบอร์ด 70
รูปที่ 5.1 จอภาพเริ่มต้นใช้งาน 71
รูปที่ 5.2 เมนเมนู 71
รูปที่ 5.3 เมนูย่อยของโปรแกรม 72
รูปที่ 5.4 โปรแกรมอีดิเตอร์ 72
รูปที่ 5.5 การเขียนแอสเซมบลี 73
รูปที่ 5.6 การเขียนโปรแกรมผิดไวยากรณ์ 73
รูปที่ 5.7 การเขียนโปรแกรมโดยใช้คำสั่งผิด 74
รูปที่ 5.8 แสดงภาพขณะทำการรันโปรแกรม 74
รูปที่ 5.9 แสดงภาพขณะทำการอันแอสแซมเบลอร์ 75
รูปที่ 5.10 เมนูย่อยของคำสั่ง MEM&REG 76
รูปที่ 5.11 คำสั่งในการอ่านข้อมูลหน่วยความจำโปรแกรม 76
รูปที่ 5.12 ผลของการอ่านโปรแกรมที่แอดเดรส 8001H 77
รูปที่ 5.13 เมนูย่อยของคำสั่งเกี่ยวกับ หน่วยความจำข้อมูล 77
รูปที่ 5.14 ทำการเขียนข้อมูลลงบนหน่วยความจำข้อมูล 78
รูปที่ 5.15 ทำการอ่านข้อมูลจากหน่วยความจำข้อมูล 78

รูปที่ 5.16 แสดงการคัดลอกข้อมูลโดยใช้คำสั่ง MOVE 78
รูปที่ 5.17 แสดงการเปรียบเทียบข้อมูลที่ทำการคัดลอก 79
รูปที่ 5.18 แสดงการเติมข้อมูลโดยใช้ คำสั่ง FILL 79
รูปที่ 5.19 แสดงการหาข้อมูลโดยใช้คำสั่ง FIND 80
รูปที่ 5.20 เมนูย่อยของคำสั่งที่ใช้ในการติดต่อกับพีซี 80
รูปที่ 5.21 การตั้งค่าความเร็วในการติดต่อทางพอร์ทอนุกรม 81
รูปที่ 5.22 การรันโปรแกรมทางพอร์ทขนานจากพีซี 81
รูปที่ 5.23 แสดงสถานะของการรับโปรแกรมจากพีซีผ่านทางพอร์ทอนุกรม 81
รูปที่ 5.24 เมนูย่อยของการส่งข้อมุลออกจากพอร์ทของผู้ใช้ 82
รูปที่ 5.25 การรับข้อมูลจากพอร์ทของผู้ใช้ 82
รูปที่ 5.26 การส่งข้อมูลออกทางพอร์ทของผู้ใช้ 82
รูปที่ 5.27 เมนูย่อย ของเครื่องมือต่าง ๆ 83
รูปที่ 5.28 แสดงสถานะการเชื่อมต่อผ่านทางพอร์ทเครื่องมือ 83
รูปที่ 5.29 เมนูเลือกเล่นดนตรี 84
รูปที่ 5.30 เพลง Happy Birth Day 84
รูปที่ 6.1 ซิงเกิลบอร์ด 88

สารบัญตาราง

ตารางที่ 2.1 แสดงชิปเดี่ยวตระกูล MCS-51 3
ตารางที่ 2.2 ตำแหน่งของรีจิสเตอร์ใช้งานทั่วไป 14
ตารางที่ 2.3 แบงค์ของรีจิสเตอร์ใช้งานทั่วไป 14

บทที่ 1
บทนำ

ในปัจจุบันนี้ เทคโนโลยีได้ก้าวหน้าไปรวดเร็วมาก คอมพิวเตอร์ได้เข้ามามีบทบาทในชีวิตประจำวันของมนุษย์มากขึ้น ซึ่งจะนำมาใช้ในการควบคุมเครื่องมือเครื่องใช้ต่างๆ คอมพิวเตอร์ที่ใช้อยู่แพร่หลายทุกวันนี้เป็นคอมพิวเตอร์แบบไมโครคอมพิวเตอร์ ซึ่งเป็นเครื่องคอมพิวเตอร์ที่เหมาะใช้งานในบ้านได้ แต่คอมพิวเตอร์ชนิดนี้ก็ยังใหญ่เกินไปที่จะนำมาควบคุม เครื่องมือเครื่องใช้ต่าง ๆ ดังนั้นจึงได้มีการพัฒนาชิปที่เรียกว่า ไมโครคอนโทรลเลอร์ (Microcontroller) ขึ้นมาซึ่งก็เป็นไมโครโปรเซสเซอร์ขนาดเล็กที่ประกอบด้วยหน่วยหลัก ๆ ของคอมพิวเตอร์ เนื่องมาจาก ไมโครคอนโทรลเลอร์มีขนาดเล็กและราคาถูก จึงถูกนำมาประยุกต์ใช้งานด้านต่าง ๆ อย่างกว้างขวาง ซึ่งไมโครคอนโทรลเลอร์ของบริษัท Intel ได้รับความนิยมมากสุด แต่โดยพื้นฐานแล้วโครงสร้างภายในของแต่ละบริษัทก็ยังมีโครงสร้างหลักที่เหมือนกัน แต่จะแตกต่างกันในการเพิ่มอุปกรณ์พิเศษเข้าไปเท่านั้น

1.1จุดประสงค์ของการจัดทำโครงงาน

จุดประสงค์ในการทำโครงงานนี้ขึ้นเพื่อศึกษาการทำงานของไมโครคอนโทรลเลอร์ และนำมาออกแบบสร้างชิงเกิลบอร์ดเพื่อใช้ในการศึกษาการทำงานของ MCS-51 โดยชิงเกิลบอร์ดที่ทำขึ้นมานี้มีคุณสมบัติดังนี้

ไมโครคอนโทรลเลอร์ 80C31 (Clock=11.0592 MHz)
หน่วยความจำข้อมูลภายใน 128 byte
หน่วยความจำโปรแกรม 32/32 Kbyte 0000-7FFFH (27256/2764)
หน่วยความจำข้อมูลภายนอก 0/32 Kbyte 0000-7FFFH (62256)
หน่วยความจำข้อมูลกับโปรแกรม 0/31 kbyte 8000-FC00H (62256)

หน่วยแสดงผล	LCD 4 แถว X 20 ตัวอักษร
	LED แสดงสถานะพร้อมการทำงาน
แป้นพิมพ์	48 ปุ่ม
จัมเปอร์	2 ทางสำหรับเลือกแบตเตอรี่ไฟสำรอง (เปิด/ปิด)
	2 ทาง สำหรับเลือกหน่วยความจำ (ภายใน/ ภายนอก)
	2 ทาง สำหรับเลือกวอทช์ดอก(Watch Dog) (ทำงาน/ไม่ทำงาน)
เสียง	ลำโพง
ส่วนเชื่อมต่อ	20 ขา พรินเตอร์พอร์ท

ส่วนเชื่อมต่อ 26 ขา 8255สำหรับผู้ใช้งาน

3 ขา พอร์ทอนุกรม (RS232)

2 ขา พอร์ทเครื่องมือ (Tool Port)

2 ขา 5V DC (In/Out)

2 ขา แจ็ค DC (9โวลท์)

ส่วนประกอบเพิ่มเติม 6242 (Real Time Clock)

เพื่อความสะดวกในการใช้งานภายในโครงงานนี้จึงได้ออกแบบให้ภายในมีโปรแกรมอิดิเตอร์(Editor) และโปรแกรมแอสแซมเบลอร์(Assembler) ภายในตัวบอร์ดทั้งหมด เพื่อผู้ใช้งานหรือผู้ที่ต้องการที่จะศึกษาไมโครคอนโทรลเลอร์สามารถใช้งาน หรือทำการทดลองได้โดยสะดวก โดยที่ไม่ต้องใช้เครื่องคอมพิวเตอร์ในการแปลงภาษาแอสแซมบลีให้เป็นภาษาเครื่อง ซึ่งก็ทำให้การใช้งานมีความคล่องตัวมากยิ่งขึ้น เพราะตัวโครงงานมีความสามารถที่ทำงานโดยลำพังได้ (Stand Alone) ซึ่งก็จะทำให้สะดวกในการใช้งาน

1.2 ประโยชน์และการประยุกต์ใช้งาน

การจัดทำโครงงานครั้งนี้ทำให้ทราบถึง การทำงานของไมโครคอนโทรลเลอร์สถาปัตยกรรมของไมโครคอนโทรลเลอร์ ตลอดจนการนำประยุกต์ไปใช้งาน ในด้านต่าง ๆ ซึ่งโครงงานชิ้นนี้ เมื่อทำเสร็จโดยสมบูรณ์ จะสามารถนำไปใช้งาน ได้อย่างกว้างขวาง เนื่องจากทรัพยากรของระบบที่ได้ทำการออกแบบไว้ เพียงพอต่อการใช้งานมากพอสมควร ซึ่งสามารถนำโครงงานชิ้นนี้ไปใช้ในการควบคุมระบบอื่น ๆ ได้อีกหลายอย่าง รวมไปถึงผู้เริ่มการศึกษาการใช้งานไมโครคอนโทรลเลอร์ ก็จะสามารถนำโครงงานชิ้นนี้ไปทำการศึกษาทดลองปฏิบัติ รวมไปถึงนำไปประยุกต์ใช้งานในด้านอื่น ๆ ได้เป็นอย่างดี

บทที่ 2

ไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์แบบชิปเดี่ยวตระกูล MCS-51 นี้ผลิตขึ้นโดยบริษัทอินเทลมีอยู่ด้วยกันหลายเบอร์ซึ่งมีรายละเอียดดังตารางที่ 2.1

ตารางที่2.1

Device	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	8-Bit I/O Ports	16-Bit Timer/ Counters	Programmable Counter Array (PCA)	UART	Serial Expansion Port (SEP)	Global Serial Channel (GSC)	DMA Channels	A/D Channels	Interrupt Sources/ Vectors	Power Down and Idle Modes
8051	8031	-	4K	128	4	2		✓					6/5	
8051AH	8031AH	8751H 8751BH	4K	128	4	2		✓					6/5	
8052AH	8032AH	8752BH	8K	256	4	3		✓					8/6	
80C51BH	80C31BH	87C51	4K	128	4	2		✓					6/5	✓
80C52	80C32	-	8K	256	4	3		✓					8/6	✓
83C51FA	80C51FA	87C51FA	8K	256	4	3	✓	✓					14/7	✓
83C51FB	80C51FB	87C51FB	16K	256	4	3	✓	✓					14/7	✓
83C152JA	80C152JA	-	8K	256	5	2		✓		✓	2		19/11	✓
-	80C152JB	-	-	256	7	2		✓		✓	2		19/11	✓
83C152JC	80C152JC	-	8K	256	5	2		✓		✓	2		19/11	✓
-	80C152JD	-	-	256	7	2		✓		✓	2		19/11	✓
83C452	80C452	87C452P	8K	256	5	2		✓					9/8	✓

2.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

-เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต

-มีวงจรออสซิลเลเตอร์และวงจรผลิตสัญญาณนาฬิกาภายในไอซี

-มีขาสัญญาณอินพุทเอาท์พุทจำนวน 32 บิต

-สามารถเชื่อมต่อหน่วยความจำข้อมูลภายนอก (External Data Memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 K

-สามารถเชื่อมต่อหน่วยความจำภายนอก (External Program Memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 K

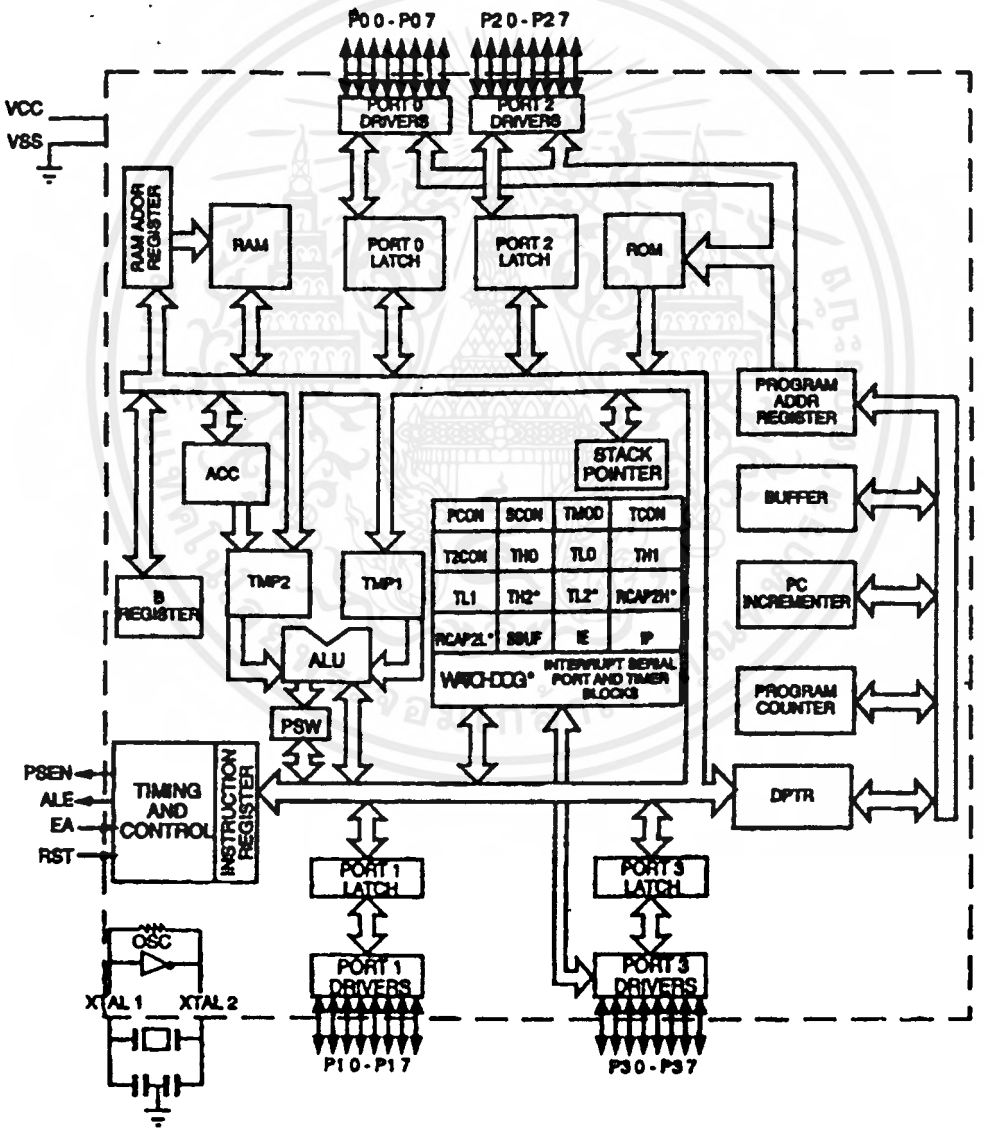
-มีหน่วยความจำโปรแกรมภายในตัว (On Chip Program Memory) ขนาด 4 K โดยเฉพาะเบอร์ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 8 K สำหรับเบอร์ 8031 และ 8032 จะไม่มีหน่วยความจำในส่วนนี้

-มีหน่วยความจำข้อมูลภายในตัว (On Chip Data Memory) ขนาด 128 ไบต์ โดยเฉพาะเบอร์ 8032 และ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 256 ไบต์

-หน่วยความจำข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย ทำให้การควบคุมหรือการตรวจสอบสถานะบิตทำได้ง่าย ส่งผลให้การเขียนโปรแกรมทำได้ง่ายมากขึ้น

-มีไทเมอร์/เคาน์เตอร์ (Timer/Counter) ขนาด 16 บิตจำนวน 2 ตัวโดยเฉพาะเบอร์ 8032 และ 8052 จะมีถึง 3ตัว

-การอินเตอร์รัปต์สามารถทำได้จาก 5 แหล่งกำเนิด โดยเฉพาะเบอร์ 8032 และ 8052 จะทำการอินเตอร์รัปต์ได้ จาก 6 แหล่งกำเนิด โดยการอินเตอร์รัปต์ยังสามารถจัดระดับความสำคัญได้เป็น 2 ระดับ



รูปที่ 2.1 บล็อคไดอะแกรมของ MCS-51

-มีพอร์ทสื่อสารอนุกรมภายในตัวเอง ซึ่งทำงานเป็นแบบฟูลดูเพล็กซ์ (full duplex)

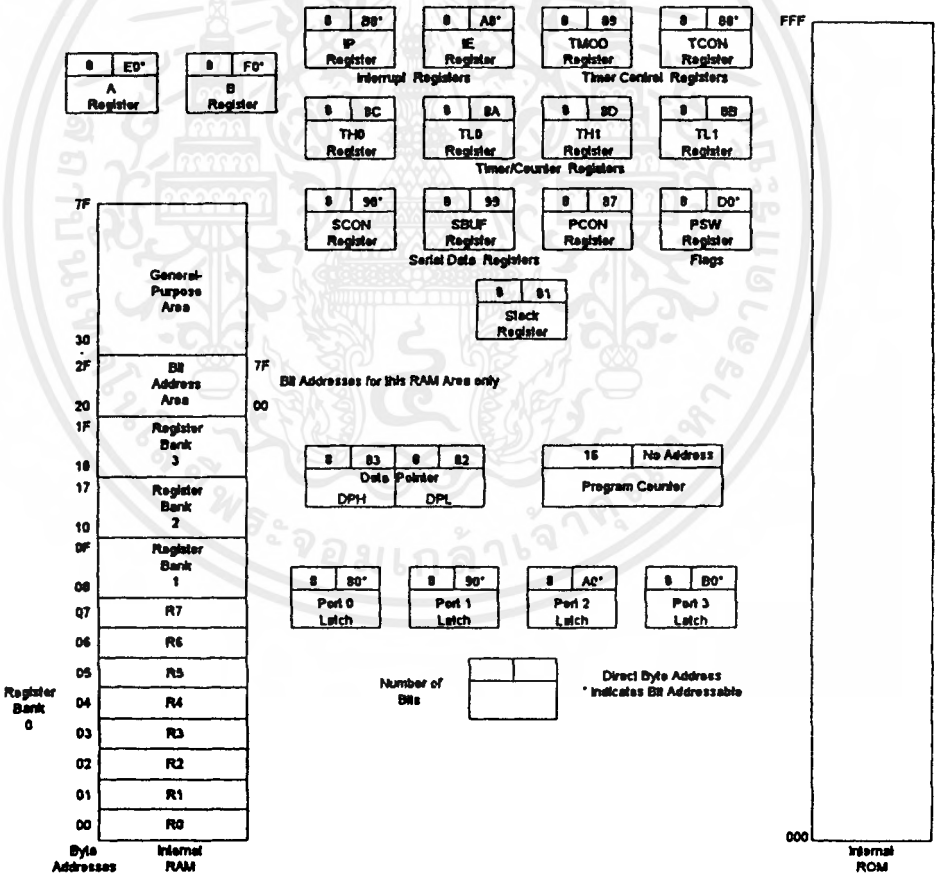
-มีคำสั่งในการคำนวณทางคณิตศาสตร์และทางตรรกศาสตร์

-คำสั่งโดยส่วนใหญ่ใช้เวลาการทำงานเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลความถี่ 12 เมกะเฮิร์ต

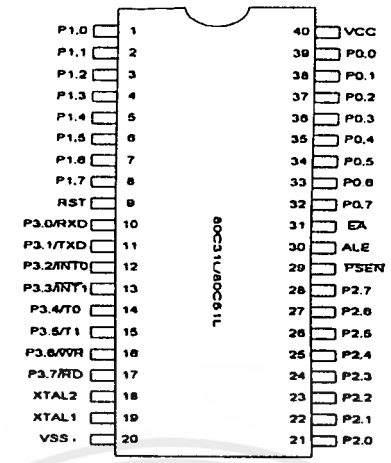
-ต้องการแหล่งจ่ายไฟ 5 โวลท์ เพียงชุดเดียว

2.2 โครงสร้างภายในของ MCS-51

MCS-51 ใช้เทคโนโลยีในการผลิตเป็นแบบ NMOS และCMOS เบอร์ 8032 และ 8052 จะมีรอมอยู่ภายในตัวจึงสะดวกสำหรับโปรแกรมเมอร์ที่จะเขียนโปรแกรมด้วยภาษาเบสิก โครงสร้างภายในสำหรับเบอร์ 8031 ดังแสดงในรูปที่ 2.1 และ 2.2



รูปที่ 2.2 ตำแน่งของรีจิสเตอร์ต่าง ๆ และหน่วยความจำภายใน



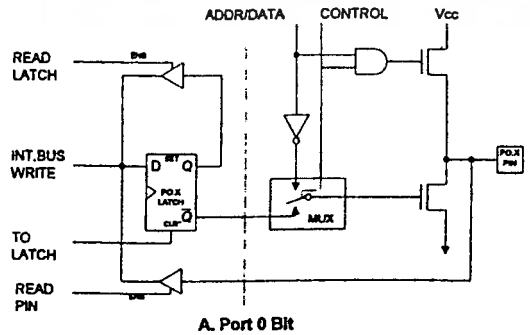
รูปที่ 2.3 การจัดวางขาของ 8031

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูปที่ 2.3 สำหรับหน้าที่การใช้งานของแต่ละขามีดังนี้

-Vcc ขาที่ 40 เป็นขาป้อนระดับแรงดันไฟเลี้ยงที่ +5 โวลท์

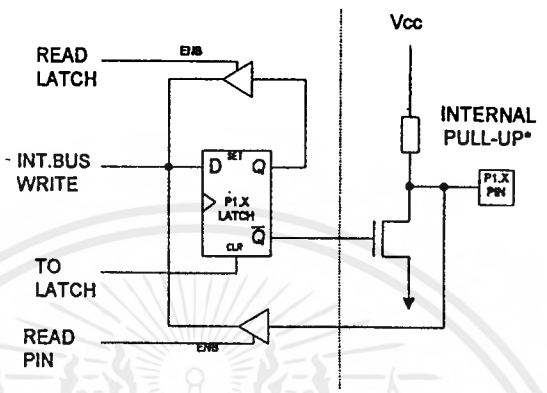
-Vss ขาที่ 20 เป็นขาที่ต้องต่อกับกราวนด์ ซึ่งจะเชื่อมต่อกับกราวนด์ของทุกอุปกรณ์

-**PORT 0** (ขา 32-39) มีทั้งหมด 8 บิตด้วยกันคือ (P0.7-P0.0) มีโครงสร้างแบบ Open Drain Bi- Directional ดังแสดงในรูปที่ 2.4 พอร์ทนี้ทำงานได้สองหน้าที่ด้วยกันคือ แอดเดรสบัส และ ดาต้าบัสเมื่อต้องการติดต่อกับหน่วยความจำภายนอกหรือเป็น ไอโอพอร์ท ถ้าต้องการให้ทำงานเป็นอินพุทต้องส่งลอจิก "1" ไปยังพอร์ทนี้ จะมีผลให้ Q ของ DFF เป็น "1" ทำให้ FET ตัวล่างมีสถานะ OFF สัญญาณที่ใช้อ่านอินพุทพอร์ทแลทช์ โดยส่งสัญญาณ Read Latch ไปกระตุ้นที่ Tri State Buffer ตัวบน และการอ่าน Port (pin) จะใช้สัญญาณ Read (pin)



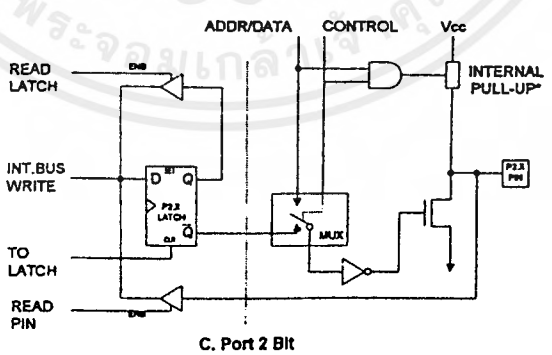
รูปที่ 2.4 โครงสร้างพอร์ท 0

-PORT 1 (ขา 1-8) มีทั้งหมด 8 บิตคือ (P1.0-P1.7) มีโครงสร้างคล้ายพยร์ท 0 แต่จะใช้ความต้านทานภายใน พูลอัพแทน (Internal Pull up Register) มีโครงสร้างดังรูปที่ 2.5



รูปที่ 2.5 โครงสร้างพอร์ท 1

-PORT 2 (ขา 21-28) มีทั้งหมด 8 บิตคือขา (P2.7-2.0) มีโครงสร้างคล้ายพอร์ท 0 โดยมีเฟตตัวล่างเดียวตัวเดียวส่วนด้านบนใช้ความต้านทานพูลอัพแทน (Internal Pull Up) พอร์ทนี้ทำงาน 2 หน้าที่ คือสามารถใช้เป็นแอดเดรสบัส ขนาด 8 บิต (A15-A8) และเป็นไอโอพอร์ทใช้งานทั่วไปเมื่อจะใช้งานเป็นอินพุทต้องส่งลอจิก "1" มาที่พอร์ทนี้ก่อนเพื่อบังคับให้เฟตอยู่ในสภาวะ Off



รูปที่ 2.6 โครงสร้างพอร์ท 2

-PORT 3 (ขา10-ขา17) มีทั้งหมด 8 บิตคือ ขา (P3.7-P3.0) มีโครงสร้างคล้ายพอร์ท 1 ทำงานได้ 2หน้าที่ คือเป็น ไอโอพอร์ทถ้าจะโปรแกรมให้เป็นอินพุทพอร์ทต้องส่งลอจิก "1" มาที่พอร์ทนี้ก่อน และอีกหน้าที่หนึ่งก็คือ ใช้ส่งสัญญาณควบคุมออกมา และรับสัญญาณเข้าไป สัญญาณต่าง ๆ มีดังนี้

P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับข้อมูลแบบอนุกรม (UART)

P3.1/TXD (Serial Output Port)เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม (UART)

P3.2/INT0 (External Interrupt 0) ใช้รับสัญญาณการขัดจังหวะแบบภายนอกเบอร์ 0

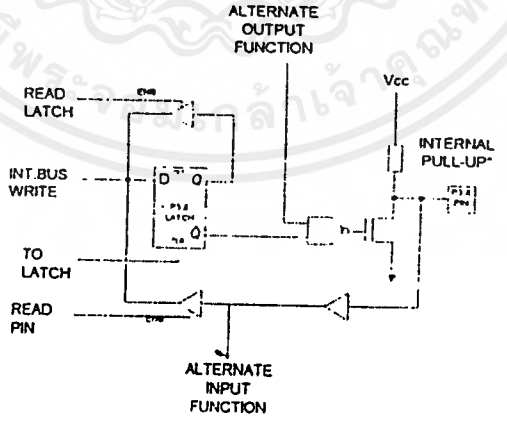
P3.3/INT1 (External Interrupt 1) ใช้รับสัญญาณการขัดจังหวะจากภายนอกเบอร์ 1

P3.4/T0 (Counter0 External Input) ขารับสัญญาณพัลส์อินพุทเข้าไปยังวงจร Counter0 (เป็นอินพุทโหมดเคาน์เตอร์)

P3.5/T1 (Counter1 External Input) เป็นขารับสัญญาณพัลซ์อินพุทเข้าไปยังวงจร Counter1 (เป็นอินพุทโหมดเคาน์เตอร์)

P3.6/WR (External Data Memory Write Strobe) เป็นขาสัญญาณควบคุมการเขียนข้อมูลลงหน่วยความจำข้อมูลภายนอก

P3.7/RD (External Data Memory Read Strobe) เป็นขาสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำข้อมูลภายนอก



รูปที่ 2.7 โครงสร้างพอร์ท 3

-RST (ขา 9) ใช้สำหรับการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสถานะเป็น 1 อย่างน้อย 2 แมชชีนไซเคิล ในขณะที่ออสซิลเลเตอร์ยังทำงานอยู่ เมื่อป้อนไฟเลี้ยง +5

โวลท์ เพื่อทำให้เกิดการรีเซตเมื่อเริ่มป้อนไฟเลี้ยง ซึ่งเรียกว่า Power On Reset การรีเซตจะให้ค่าในรีจิสเตอร์ต่างๆ เปลี่ยนไปเป็นค่าหนึ่งดังในรูปที่ 2.8

REGISTER	CONTENT
PC	000H
ACC	00H
B	00H
PSW -	00H
SP	00H
DPTR	0000H
P0-P3	0FFH
IP	00H
IE	0X000000B
TMOD	00H
TCON	00H
T2CON	00H
THO	00H
TLO	00H
TH1	00H
TL1	00H
TH2	00H
TL2	00H
RCAP2H	00H
RCAP2L	00H
SCON	00H
SBUF	INDETERMINE
IOOON	00H

รูปที่ 2.8 ค่าของรีจิสเตอร์เมื่อเกิดการรีเซท 8031

-**ALE** (ขา 30) เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแลตช์ (Latch) ค่าตำแหน่งแอดเดรสไบต์ต่ำ (Address Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่เป็น อินพุทรับพัลส์ในการโปรแกรม (Program Pulse Input) ในส่วนของหน่วยความจำ อีพรอม สัญญาณนี้จะแอคทีฟทุก ๆ 2 ครั้งใน 1 แมชชีนไซเคิล

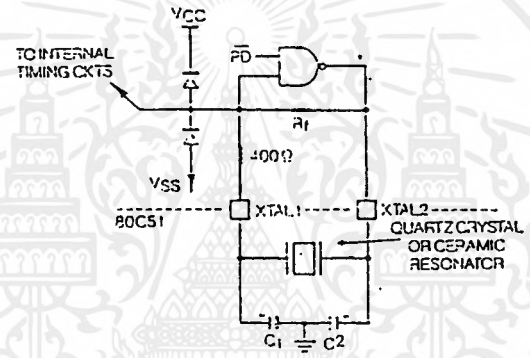
-**PSEN** (ขา 29) ทำหน้าที่เป็นสัญญาณสโตรปเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรปจำนวน 2 ครั้งในแต่ละแมชชีนไซเคิล แต่ในขณะที่ติดต่อกับหน่วยความจำภายนอกจะไม่มีการส่งสัญญาณสโตรปแต่อย่างใด

-**EA** (ขา 30) ใช้เลือกหน่วยความจำโปรแกรมภายนอก

ป้อน "0" จะอ่านโปรแกรมจากภายนอกชิป

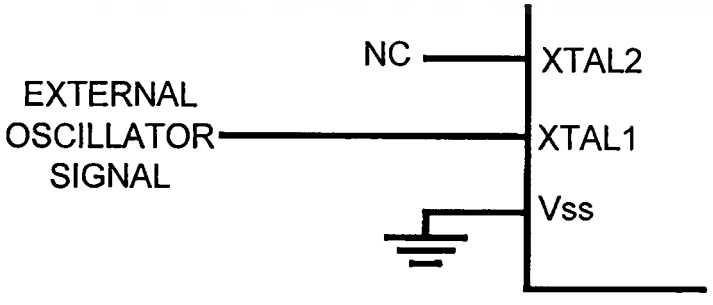
ป้อน "1" จะอ่านโปรแกรมจากภายในชิป

-XTAL 1 (ขา19) ขานี้จะต่อเข้ากับขาของวงรขยายแบบป้อนกลับเฟสสัญญาณ (Inverting Amplifier) ที่ประกอบเป็นวงจรสัญญาณออสซิลเลเตอร์ ในรูปที่ 2.6 จะเห็นว่าวงจรภายในออสซิเลเตอร์ แนนเกทจะทำหน้าที่เป็นวงจรขยายแบบกลับเฟส ของสัญญาณที่จะควบคุมให้มีการออสซิเลเตอร์หรือไม่ก็ขึ้นกับสัญญาณ PD ซึ่งต่อจากบิต PD ของรีจิสเตอร์ PCON ถ้าต้องการใช้สัญญาณนาฬิกา (Clock Signal) จากภายนอกมาเป็นสัญญาณนาฬิกา ควบคุมการทำงานของ 8031 ก็ให้ป้อนสัญญาณเข้ามาที่จุดนี้ แต่ถ้าต้องการใช้วงจรออสซิลเลเตอร์ภายในก็ให้ต่อคริสตอล หรือเซรามิคเรโซเนเตอร์ ดังรูปที่ 2.9 คาปาซิเตอร์ในวงจรควรมีค่าประมาณ 20 pF



รูปที่ 2.9 วงจรออสซิลเลเตอร์ภายใน 8031

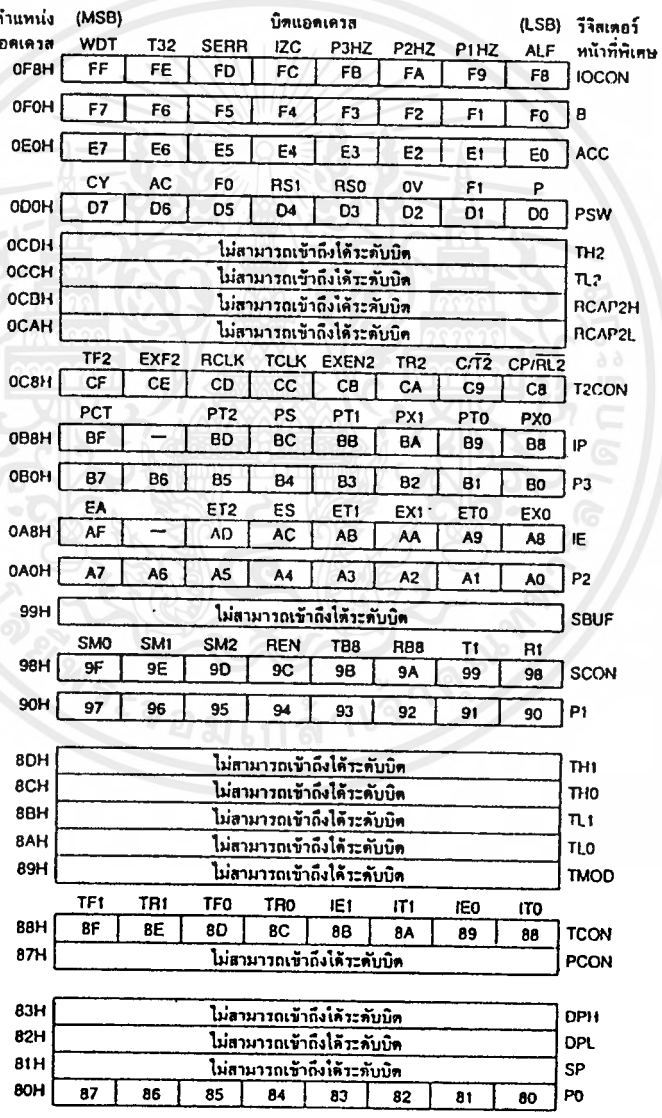
-XTAL 2 (ขา 18) ขานี้เป็นวงจรเอาท์พุทของวงจรขยายแบบกลับเฟสสัญญาณที่ประกอบเป็นวงจรออสซิลเลเตอร์ (อินพุทคือขา XTAL 1) ถ้าจะใช้สัญญาณนาฬิกาที่สร้างมาจากสัญญาณภายนอกมาเป็นสัญญาณนาฬิกาของ 8031 แล้ว ให้ปล่อยขานี้ลอยไว้แล้วป้อนสัญญาณนาฬิกาจากภายนอกเข้ามาที่ขา XTAL 1 ดังรูปที่ 2.10



รูปที่ 2.10 8031ที่ทำงานโดยสัญญาณนาฬิกาที่มาจากภายนอก

2.3 รีจิสเตอร์ของ 8031

จากแผนภาพในรูปที่ 2.1 แสดงให้เห็นโครงสร้างภายในของไมโครคอนโทรลเลอร์ 8031 ซึ่งประกอบด้วยหน้าที่การทำงานต่างๆ ภายในไอซี 8031 จำนวนมาก โดยแต่ละบล็อกซึ่งเป็นวงจร ควบคุมรีจิสเตอร์หรือหน่วยความจำภายในของ 8031 จะถูกเชื่อมต่อเข้าด้วยกัน ผ่านทางเส้นสัญญาณที่เรียกว่าบัสข้อมูลภายใน รีจิสเตอร์และหน่วยความจำเหล่านี้ จะถูกนำไปใช้ในระหว่างการประมวลผลคำสั่ง หน้าที่ของโปรแกรมที่ผู้ใช้สร้างขึ้นมาก็เป็นการควบคุมการรับหรือส่งข้อมูลระหว่างรีจิสเตอร์เหล่านี้



รูปที่ 2.11 ตำแหน่งของรีจิสเตอร์หน้าที่พิเศษต่าง ๆ

ในความเป็นจริงแล้วรีจิสเตอร์อาจจะพิจารณาได้ว่าเป็นหน่วยความจำแรมประเภทหนึ่งที่อยู่ภายในไมโครคอนโทรลเลอร์ และสามารถนำข้อมูลเข้าไปเก็บไว้ภายในรีจิสเตอร์หรืออ่านข้อมูลนี้ออกมาในภายหลังได้เช่นเดียวกับหน่วยความจำ ความแตกต่างระหว่างหน่วยความจำและรีจิสเตอร์ที่สำคัญเป็นลักษณะของการติดต่อกับหน่วยประมวลผลกลาง โดยหากว่าเป็นการติดต่อกับรีจิสเตอร์จะใช้ชุดคำสั่งเกี่ยวกับชื่อของรีจิสเตอร์นั้นโดยตรงไม่จำเป็นต้องให้ความสนใจว่าตำแหน่งของรีจิสเตอร์นั้น จะอยู่ที่ไหนเป็นลักษณะที่ตรงกันข้ามกับหน่วยความจำภายใน ซึ่งจะต้องระบุถึงตำแหน่งโดยชัดเจนนอกจากนั้นแล้วอาจมีชุดคำสั่งบางอย่างที่สามารถดำเนินการได้กับเฉพาะรีจิสเตอร์ท่านั้น หากจะจำแนกกลุ่มรีจิสเตอร์ตามลักษณะ ของการนำไปใช้งานนั้นจะแยกออกได้เป็น 3 ประเภท คือ รีจิสเตอร์ที่เกี่ยวกับแอดเดรส รีจิสเตอร์ใช้งานทั่วไป และ รีจิสเตอร์ฟังก์ชันหน้าที่พิเศษ

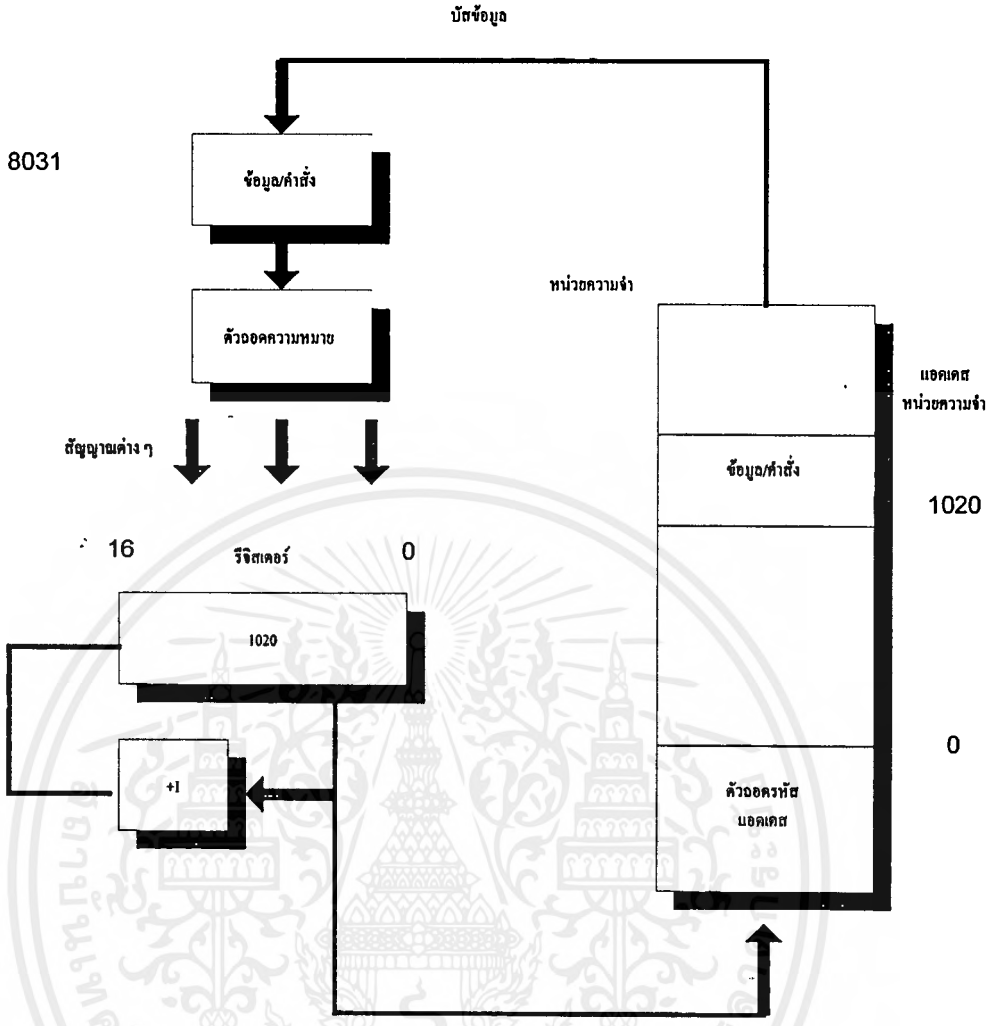
2.3.1 รีจิสเตอร์ที่เกี่ยวกับแอดเดรส (Address Registers)

รีจิสเตอร์กลุ่มนี้เป็นรีจิสเตอร์ขนาด 16 บิต ที่จะใช้งานเพื่อเก็บข้อมูลแอดเดรส เป็นสำคัญ โดยค่าที่อยู่ภายในแอดเดรสนี้ จะนำไปเป็นค่าของข้อมูลที่ส่งออกไปทางบัสแอดเดรสของไมโครคอนโทรลเลอร์ เพื่อบอกตำแหน่งที่ต้องการจะติดต่อ รีจิสเตอร์ที่ประกอบอยู่ในกลุ่มนี้ประกอบด้วย

-โปรแกรมเคาน์เตอร์ (Program Counter) เป็นรีจิสเตอร์ที่ใช้ในการชี้ตำแหน่งของหน่วยความจำโปรแกรมซึ่งจะต้องนำคำสั่งมาประมวลผลในลำดับต่อไป โดยเมื่อถึงเวลาที่ 8031 ต้องการคำสั่งที่เก็บไว้ภายในหน่วยความจำแล้ว รีจิสเตอร์นี้ออกมายังแอดเดรสบัส ซึ่งหน่วยความจำที่ต่ออยู่กับบัสนี้ก็จะรับทราบ และนำค่าที่เก็บไว้ในตัวของหน่วยความจำ ในตำแหน่งที่ต้องการส่งมาให้กับ 8031 ภายหลังจากนี้แล้วค่าของข้อมูลภายในรีจิสเตอร์นี้จะเพิ่มโดยอัตโนมัติเพื่อบอกตำแหน่งของคำสั่งถัดไป ดูแผนภาพการทำงานในรูปที่ 2.9 การใช้งานภายในโปรแกรมเรียกว่า รีจิสเตอร์ PC

- สแต็กพอยน์เตอร์ (Stack Pointer) เป็นรีจิสเตอร์ขนาด 8 บิต ทำหน้าที่เก็บตำแหน่งแอดเดรสหน่วยความจำของบริเวณที่ใช้งานเป็นสแต็ก สำหรับเก็บข้อมูลแอคคิวมูเลเตอร์ รีจิสเตอร์ต่าง ๆ รวมทั้งข้อมูลจากโปรแกรม โดยปกติแล้ว เมื่อทำการเริ่มต้นระบบใหม่ภายหลังจากเริ่มจ่ายไฟหรือมีการรีเซตขึ้น ค่าภายในสแต็กพอยน์เตอร์ จะเป็นค่า 07 H ซึ่งเป็นแอดเดรสภายในเนื้อที่ 128 ไบต์แรกของหน่วยความจำข้อมูลภายใน การใช้งานโปรแกรมเรียกว่า รีจิสเตอร์ SP

- ดาต้าพอยน์เตอร์ (Data Pointer)เป็นรีจิสเตอร์ขนาด 16 บิต ซึ่งเรียกว่า รีจิสเตอร์ DPTR และสามารถใช้งานแยกออกเป็น รีจิสเตอร์ขนาด 8 บิตจำนวน 2 ตัวคือ รีจิสเตอร์ DPH และ DPL เพื่อเก็บค่าของแอดเดรสของหน่วยความจำหรืออุปกรณ์ที่จะต้องใช้งานภายในโปรแกรม หารใช้รีจิสเตอร์ DPTR นี้นับว่ามีประโยชน์เนื่องจากทำให้หน่วยประมวลผลกลางสามารถใช้เทคนิคของการแอดเดรสแบบทางอ้อมได้



รูปที่ 2.12 แสดงการทำงานของรีจิสเตอร์โปรแกรมเคาน์เตอร์

2.3.2 รีจิสเตอร์ใช้งานทั่วไป (General Purpose Register)

รีจิสเตอร์ในกลุ่มนี้จัดเป็นพื้นที่ของหน่วยความจำที่ใช้ในการสนับสนุนการประมวลผลการทำงานจากหน่วยประมวลผลทางคณิตศาสตร์และลอจิก (ALU) เพื่อให้สามารถจัดการข้อมูลได้เร็วที่สุดนอกจากนี้ในกรณีที่ไม่ได้ใช้คำสั่งเหล่านี้ก็ยังเป็นการเก็บข้อมูลตัวแปรภายในโปรแกรมได้รีจิสเตอร์ในกลุ่มนี้มีเพียง 8 บิตมีชื่อที่เรียกว่า รีจิสเตอร์ R0,R1,R2,R3,R4,R5,R6 และR7 ซึ่งตำแหน่งของรีจิสเตอร์เหล่านี้อยู่ภายในบริเวณของหน่วยความจำข้อมูลภายใน (Internal RAM) ช่วงบริเวณแอดเดรส 00H ถึง 07H โดยจำแนกออกเป็นกลุ่ม (หรือ แบงก์) ของกลุ่มรีจิสเตอร์ ดังตารางต่อไปนี้

ตารางที่2.2

แอดเดรส	รีจิสเตอร์แบงก์	ชื่อรีจิสเตอร์ใช้งาน
00-07H	0	R0-R7
03-0FH	1	R0-R7
10-17H	2	R0-R7
18-1FH	3	R0-R7

จะเห็นได้ว่าชื่อของรีจิสเตอร์ไม่ว่าจะอยู่ในรีจิสเตอร์แบงก์ใด ก็จะมีชื่อ R0 ถึง R7 เหมือนกันทั้งสิ้น ดังนั้นการใช้งานผู้ใช้ต้องให้ความระมัดระวัง ต้องการรีจิสเตอร์นั้น ๆ จากแบงก์ใดซึ่งการกำหนดเลือกแต่ละกลุ่มของรีจิสเตอร์นี้ก็ทำได้โดยง่ายเพียงการกำหนดค่าของบิตที่อยู่ภายในรีจิสเตอร์ PSW เท่านั้นตามตารางต่อไปนี้

ตารางที่2.3

รีจิสเตอร์	บิต RS0	บิต RST	ตำแหน่งหน่วยความจำ
แบงก์0	0	0	0000H
แบงก์1	0	1	0008H
แบงก์2	1	0	0010H
แบงก์3	1	1	0018H

อย่างไรก็ตามโดยทั่วไปมักจะมีการใช้งานรีจิสเตอร์ R0-R7 เฉพาะในแบงก์ 0 เท่านั้น ดังนั้นพื้นที่ของแบงก์อื่น ๆ ที่เหลือก็สามารถนำมาใช้ในลักษณะของหน่วยความจำ RAM ปกติ

2.3.3. รีจิสเตอร์หน้าที่พิเศษ (Special Function Register)

รีจิสเตอร์หน้าที่พิเศษ (SFR) เป็นรีจิสเตอร์สำหรับการควบคุมหน้าที่ และการทำงานของอุปกรณ์หรือพอร์ทของ 8031 ทั้งหมด ตำแหน่งของรีจิสเตอร์เหล่านี้จะจัดอยู่ในบริเวณแอดเดรส 80H-FFH การใช้งานรีจิสเตอร์หน้าที่พิเศษเหล่านี้ สามารถทำได้ทั้งการระบุถึงชื่อรีจิสเตอร์หรือตำแหน่งของแอดเดรสที่เป็นของรีจิสเตอร์นั้นก็ได้ ในรูปที่ 2.10 ได้แสดงถึงการจัดพื้นที่หน่วยความ

จำสำหรับรีจิสเตอร์หน้าที่พิเศษเหล่านี้ โดยมีข้อสังเกตว่ารีจิสเตอร์ที่อยู่ในตำแหน่งแอดเดรสที่เป็นจำนวนทวีคูณของค่า 8 จะสามารถอ้างถึงในระดับบิตได้ด้วย (นั่นคือแอดเดรส 80H 88H 90H A0H B0H D0H E0H และ F0H)

แอคคิวมูเลเตอร์ (Accumulator) หรือ ACC เป็นรีจิสเตอร์ขนาด 8 บิต ทำหน้าที่ในการเก็บข้อมูลที่จะส่งให้กับหน่วยทำงานภายในซีพียู และเก็บผลลัพธ์ที่ได้จากการทำงานนั้น การทำงานของรีจิสเตอร์นี้ มีลักษณะเช่นเดียวกับตัวแอคคิวมูเลเตอร์ของโปรเซสเซอร์ทั่วไป การทำงานภายในของโปรแกรมจะเรียกว่า รีจิสเตอร์ A

รีจิสเตอร์ B เป็นรีจิสเตอร์ที่ใช้สำหรับการทำคำสั่งการคูณและการหารตัวเลข ในกรณีที่ไม่ใช้ในการคำนวณทางด้านคณิตศาสตร์ ก็สามารถนำไปใช้งานเช่นเดียวกับรีจิสเตอร์ทั่วไปได้

โปรแกรมสเตตัสเวิร์ด (PSW) รีจิสเตอร์นี้ทำหน้าที่บอกถึงแฟลก ซึ่งเป็นตัวแสดงสภาวะการทำงานต่างๆ รวมทั้งบิตสำหรับการกำหนดเลือกแบงก์ ของรีจิสเตอร์ที่ใช้งานด้วย ดังแสดงในรูปที่ 2.13

ชื่อบิต PSW ตำแหน่ง DOH ค่ากำหนดเริ่มต้น 0000 0111

CV	AC	F0	RS1	RS0	0V	-	P

ชื่อบิต	ตำแหน่ง	ความหมาย
CY	PSW.7	แฟล็กทด (Carry Flag)
AC	PSW.6	แฟล็กช่วยทด (Auxilary Carry Flag)
F0	PSW.5	แฟล็กที่ผู้ใช้สามารถนำไปใช้งานได้โดยอิสระ
RS1	PSW.4	บิตสำหรับการเลือกแบงก์ของรีจิสเตอร์บิต 1
RS2	PSW.3	บิตสำหรับการเลือกแบงก์ของรีจิสเตอร์บิต 0
OV	PSW.2	แฟล็กโอเวอร์โฟลว์ (Overflow Flag)
-	PSW.1	
P	PSW.0	แฟล็กพาริตี้ (Parity Flag) ใช้กับแอกคิวมูเลเตอร์

รูปที่ 2.13 บิตต่างๆภายในรีจิสเตอร์ PSW (Program Status Word)

รีจิสเตอร์ที่เกี่ยวข้องกับพอร์ท (Port Register)รีจิสเตอร์เหล่านี้จะมีความเกี่ยวข้องกับการทำงานของพอร์ทอินพุท เอาท์พุทโดยตรง ซึ่งแต่ละตัวรีจิสเตอร์ขนาด 8 บิต สามารถใช้งานได้ทั้งลักษณะของการอินพุทหรือการเอาท์พุทข้อมูลได้ การดำเนินการใด ๆ ที่เกี่ยวข้องกับพอร์ททั้งสี่นี้จะมีผลให้ข้อมูลที่ตำแหน่งของพอร์ทเหล่านี้เปลี่ยนแปลงไปเช่นกัน นอกจากนี้ พอร์ท P0 และP2 ยังสามารถนำมาใช้ในการติดต่อกับหน่วยความจำโปรแกรม หรือหน่วยความจำข้อมูลภายนอกได้ โดย

พอร์ท P2 จะเป็นค่าของแอดเดรส 8 บิตบนของหน่วยความจำ และพอร์ท P0 นั้นในช่วงเริ่มแรกจะเป็น ค่าของแอดเดรส 8 บิตล่างของหน่วยความจำ ช่วงเวลาต่อมาจึงนำเอาพอร์ท P0 ไปใช้เป็นบัสสำหรับการรับหรือส่งข้อมูลกับหน่วยอุปกรณ์ภายนอก สำหรับ พอร์ท P3 นั้นนอกเหนือจากจะใช้ในฐานะของพอร์ทอินพุท เอาท์พุท เช่นปกติ แล้วยังนำมาใช้ในฐานะของบัสควบคุมเกี่ยวกับสัญญาณอินเตอร์รัปต์ได้อีกด้วย

รีจิสเตอร์ SBUF เป็นบัฟเฟอร์ขนาด 8 บิต สำหรับการสื่อสารข้อมูลทั้งการรับและการส่งข้อมูล ซึ่งตามความเป็นจริงแล้วบัฟเฟอร์นี้มีอยู่ด้วยกัน 2 ชุด และแยกจากกันอย่างชัดเจนสำหรับการส่งและการรับ โดย ซีพียูจะทำการเลือกบัฟเฟอร์ที่เหมาะสมโดยอัตโนมัติ

รีจิสเตอร์ PCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมหน้าที่การทำงานในสามลักษณะ ซึ่งได้แก่ การควบคุมการทำงานของโปรเซสเซอร์ (บิต IDL และ PD) การกำหนดอัตราการทวีคูณของอัตราเร็วในการสื่อสารข้อมูลอนุกรม (บิต SMOD) และแฟลกสภาวะสำหรับการใช้งานทั่วไป (บิต GRO และ GRI) ดังได้แสดงในรูปที่2.14

-บิต PD (Power down) เป็นการกำหนดให้ลดกำลังไฟฟ้าที่จ่ายให้กับส่วนของหน่วยภายในโปรเซสเซอร์ลง โดยยังคงมีกำลังไฟฟ้าจ่ายให้กับส่วนหน่วยความจำข้อมูลภายในผ่านทางสัญญาณ RST วิธีนี้มักมาใช้ในกรณีที่มีการตรวจสอบการไม่มีกำลังไฟฟ้า (Power Failure) โดยวงจรตรวจสอบภายนอกจะต้องมีการอินเตอร์รัปต์เข้ามา เพื่อทำการเก็บข้อมูลที่กำลังประมวลผลอยู่ก่อนและเมื่อมีกระแสไฟฟ้าจ่ายให้เป็นปกติแล้ว จึงค่อยนำข้อมูลนั้นมาประมวลผลต่อไป

-บิต IDL (Idle Mode) เป็นการกำหนดให้โปรเซสเซอร์หยุดการทำงานชั่วขณะและจะกลับมาอยู่ในสภาพปกติอีกครั้ง เมื่อทำการรีเซตทางฮาร์ดแวร์ หรือมีการอินเตอร์รัปต์อย่างใดอย่างหนึ่งเกิดขึ้น การทำงานในลักษณะนี้สามารถเกิดขึ้นได้ ก็เนื่องจากว่าสภาวะการหยุดทำงานชั่วขณะนั้นเป็นเพียงการห้ามไม่ให้มีสัญญาณนาฬิกาจ่ายให้ส่วนของโปรเซสเซอร์เท่านั้น ส่วนของวงจรการอินเตอร์รัปต์พอร์ทอนุกรม และวงจรนับ/จับเวลา ยังคงมีสัญญาณนาฬิกาอยู่ปกติ

ชื่อบิต PCON ตำแหน่ง 97H ค่ากำหนดเริ่มต้น 0XXX 0000

SMCD	-	-	-	GF1	GF0	PD	IDL

ชื่อบิต	ตำแหน่ง	ความหมาย
SMOD	PCON.7	บิตกำหนดการทวีคูณของอัตราบอดปกติ
	PCON.6	
	PCON.5	
	PCON.4	
GF1	PCON.3	แฟล็กสำหรับใช้งานทั่วไปจากผู้ใช้งาน Flag 0
GF0	PCON.2	แฟล็กสำหรับใช้งานทั่วไปจากผู้ใช้งาน Flag 1
PD	PCON.1	บิตสำหรับการกำหนด Power down
IDL	PCON.0	บิตสำหรับการกำหนด Idle Mode

รูปที่ 2.14 บิตต่าง ๆภายในรีจิสเตอร์ PCON (Power Control Register)

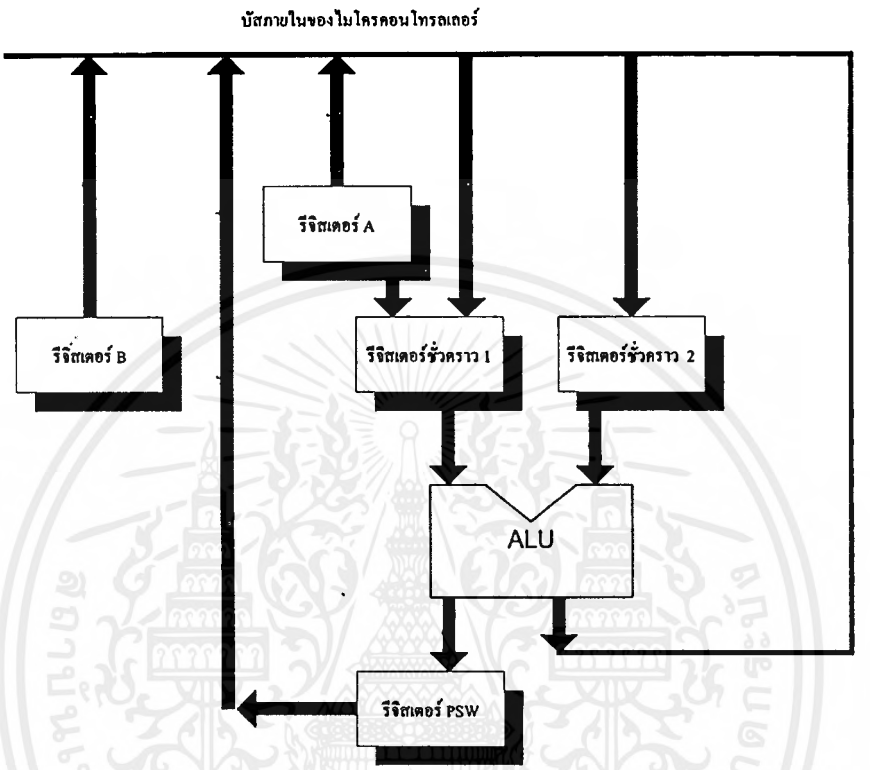
รีจิสเตอร์ IP , IE , TMOD , SCON เป็นกลุ่มของรีจิสเตอร์ที่ทำหน้าที่กำหนดการควบคุม และการทำงานของการอินเตอร์รัปต์ต่าง ๆ

2.4 หน่วยประมวลผลทางคณิตศาสตร์และลอจิก

ในการทำชุดคำสั่งที่เกี่ยวข้องกับการคำนวณทางคณิตศาสตร์ หรือการดำเนินการแบบลอจิกนั้น 8031 จะดำเนินการโดยใช้หน่วยประมวลผลทางคณิตศาสตร์และลอจิก (Arithmatic and Logic Unit) หรือที่เรียกกันว่า ALU ซึ่งจะมีรีจิสเตอร์ที่เกี่ยวข้องด้วยคือ รีจิสเตอร์เก็บข้อมูลชั่วคราว (Temporary Register) เพื่อเก็บข้อมูลตัวเลขที่จะมาดำเนินการนั้น (ดูในรูปที่ 2.13) ดังนั้นเมื่อเราได้คำสั่งที่ต้องดำเนินการโดย ALU ก็มักจะต้องส่งค่าของข้อมูลมาทางรีจิสเตอร์ A และรีจิสเตอร์อื่น ๆ ด้วยเสมอ จากนั้นหน่วยประมวลผลกลางของ8051จะเป็นตัวดำเนินการเพื่อย้ายข้อมูลเข้าไปยังรีจิสเตอร์เก็บข้อมูลชั่วคราวนี้โดยอัตโนมัติและเมื่อเสร็จสิ้นการดำเนินการแล้ว จะนำผลลัพธ์ที่ได้ออกมาเก็บไว้ที่รีจิสเตอร์ A เพื่อให้ผู้ใช้ทำการประมวลผลในโปรแกรมต่อไป นอกจากนี้ 8031 ยังมีคำสั่งที่เกี่ยวข้องกับการคูณหรือการหารด้วย ซึ่งมีการดำเนินการที่เหมือนกันเพียงแต่จะต้องนำข้อมูลมาไว้ในรีจิสเตอร์ A และ รีจิสเตอร์ B เท่านั้น

รีจิสเตอร์อีกตัวหนึ่งที่ข้อมูลของมันเป็นผลมาจากการทำงานของหน่วย ALU คือรีจิสเตอร์ PWS ซึ่งทำหน้าที่เก็บสภาวะที่เกี่ยวข้องกับผลลัพธ์จากการดำเนินการทางคณิตศาสตร์หรือทางลอจิก

ซึ่งเราสามารถพิจารณาได้ว่าเป็นกลุ่มของแฟลก เพื่อบอกผลการทำคำสั่ง ตัวอย่างเช่น หารบวกค่าตัวเลขและมีการทดเกิดขึ้น ก็จะมีแฟลกทด (Carry Flag) ที่มีสภาวะตามการทดตัวเลขนั้น เป็นต้น



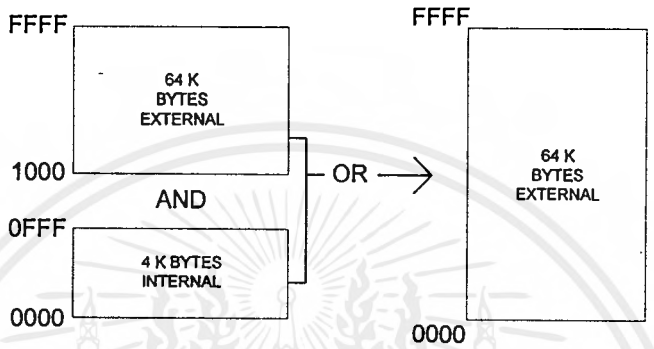
รูปที่ 2.15 แผนภาพแสดงหน่วยประมวลผลทางคณิตศาสตร์ (ALU)

2.5 หน่วยความจำของ 8031

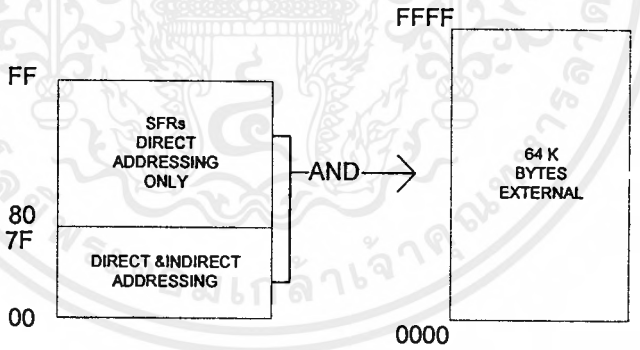
ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบ่งชนิดหน้าที่ของหน่วยความจำออกเป็น 2 ส่วน คือหน่วยความจำโปรแกรม (Program Memory) และหน่วยความจำข้อมูล (Data Memory)

หน่วยความจำโปรแกรมจะใช้สำหรับเก็บโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ซึ่งบางเบอร์จะมีหน่วยความจำในส่วนนี้อยู่ภายในตัว โดยอาจมีขนาดไม่เท่ากันหรือเป็นหน่วยความจำต่างชนิดกัน เช่น บางเบอร์เป็นรอมและบางเบอร์อาจเป็นอีพรอม และบางเบอร์อาจไม่มีหน่วยความจำในส่วนนี้เลย โปรแกรมการทำงานจะถูกเก็บไว้ยังหน่วยความจำโปรแกรมภายนอกทั้งหมด

สำหรับหน่วยความจำข้อมูลจะใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่าง ๆ จากการทำงานของโปรแกรม ซึ่งใน MCS-51 ทุกเบอร์จะมีหน่วยความจำในส่วนนี้อยู่จำนวนหนึ่งแต่อาจมีขนาดมากน้อยต่างกันไปในแต่ละเบอร์ สำหรับการจัดโครงสร้างของหน่วยความจำทั้งในส่วนของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลแสดงไว้ดังในรูปที่2.16



รูปที่ 2.16 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับ 8031



รูปที่ 2.17 ผังหน่วยความจำสำหรับเก็บข้อมูลสำหรับ 8031

2.5.1 หน่วยความจำโปรแกรม (Program Memory)

หน่วยความจำโปรแกรมของ 8031 เป็นบริเวณหน่วยความจำ สำหรับเก็บข้อมูลและคำสั่งใช้งานต่าง ๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบ ข้อมูลเหล่านี้ยังคงอยู่ไม่สูญหาย โครงสร้างของหน่วยความจำโปรแกรม มีลักษณะเช่นเดียวกับความจำที่บรรจุในไอซีหน่วยความจำประเภทต่าง ๆ เช่น หน่วยความจำแบบ ROM (Read Only Memory) หรือ EPROM (Erasable

Programmable Read Only Memory) เป็นต้น 8031 สามารอ่านข้อมูลหน่วยความจำโปรแกรมนี้ได้สูงสุดไม่เกิน 64 กิโลไบต์ และหน่วยความจำโปรแกรมสามารถแบ่งออกได้เป็น 2 ส่วนคือ หน่วยความจำโปรแกรมภายใน และ หน่วยความจำโปรแกรมภายนอก หน่วยความจำโปรแกรมภายในถูกเลือกใช้งาน ถ้าขาสัญญาณ EA มีค่าเป็น 1 โดยจะถูกใช้งานในช่วงแอดเดรส 0-0FFFH (หรือช่วงแอดเดรส 0-1FFFH ในเบอร์ 8032) นอกเหนือจากช่วงแอดเดรสนี้จะใช้หน่วยความจำโปรแกรมภายนอกทั้งหมด ในกรณีตรงกันข้ามถ้าขาสัญญาณ EA มีค่าเป็น 0 ในช่วงแอดเดรส 0-0FFFH (หรือช่วงแอดเดรส 0-1FFFH ในเบอร์ 8032) จะถูกใช้หน่วยความจำภายนอก หรือกล่าวได้ว่าถ้าขาสัญญาณ EA มีค่าเป็น 0 จะเป็นการเลือกใช้หน่วยความจำโปรแกรมภายนอกทั้งหมดตลอดช่วงแอดเดรส ไมโครคอนโทรลเลอร์เบอร์ต่าง ๆ ตระกูล 8031 นี้ สามารถขยายให้ใช้งานในหน่วยความจำภายนอกได้ทั้งสิ้น โดยกรณีที่มีหน่วยความจำโปรแกรมภายนอกอยู่แล้ว การอ้างตำแหน่งแอดเดรสที่มีทั้งหน่วยความจำภายในและภายนอกนั้น จะต้องทำการควบคุม ระดับลอจิกของสัญญาณ EA ในระดับนั้นด้วย

2.5.2 หน่วยความจำข้อมูล (Data Memory)

หน่วยความจำข้อมูลสามารถแบ่งออกได้เป็น 2 ส่วนคือ หน่วยความจำข้อมูลภายในและหน่วยความจำข้อมูลภายนอก สำหรับหน่วยความจำข้อมูลภายในยังแบ่งออกได้เป็น 2 ส่วน ย่อยคือ ส่วนที่ใช้เก็บข้อมูลทั่วไปและส่วนที่ใช้เป็นรีจิสเตอร์หน้าที่พิเศษหรือ SFR (Special Function Register) โดยส่วนที่ใช้เก็บข้อมูลทั่วไปจะถูกใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่าง ๆ จากการทำงานของโปรแกรม ส่วนรีจิสเตอร์หน้าที่พิเศษจะถูกใช้งานเป็น รีจิสเตอร์ควบคุมการทำงานและบอกสถานะการทำงานของไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์ เป็นอย่างน้อย และบางเบอร์อาจมีถึงขนาด 256 ไบต์

2.6 เวลาของการประมวลผลชุดคำสั่ง

เวลาการประมวลผลของชุดคำสั่งจนเสร็จสิ้นของ 8031 นั้นจะนับเป็นหน่วยของแมชชีนไซเคิล ซึ่งจากข้อมูลจำนวนแมชชีนไซเคิลของแต่ละคำสั่งมีค่าที่แตกต่างกันไป โดยค่าเวลา 1 แมชชีนไซเคิลนั้นจัดว่าเป็นช่วงเวลาที่น้อยที่สุดในการทำคำสั่งใดคำสั่งหนึ่ง ซึ่งหากว่า เป็นคำสั่งที่ซับซ้อนมากก็ต้องใช้เวลานาน สองถึงสามแมชชีนไซเคิล

การคำนวณหาว่าเวลาที่ใช้ในการทำคำสั่งใดจนเสร็จสิ้น จะต้องดูคำสั่งนั้นใช้จำนวนแมชชีนไซเคิลเป็นเท่าไรในการประมวลผลเวลาที่ใช้จะคำนวณตามสูตร

$$T = \frac{C \times 12}{\text{Crystal Frequency}}$$

โดย C เป็นจำนวนแมชชีนไซเคิลของคำสั่ง

Crystal Frequency เป็นค่าความถี่ของแมชชีนไซเคิลที่ใช้กับ 8031

2.7 การทำงานของ 8031

คอมพิวเตอร์จะทำงานด้วยวงจรที่เรียกว่า ฮาร์ดแวร์ ประกอบขึ้นมาเพียงอย่างเดียวไม่ได้จะต้องมีการโปรแกรมหรือคำสั่งที่จัดเรียงกันไว้ให้คอมพิวเตอร์ทำงานตามลำดับ ใน 8031 ก็เช่นเดียวกัน ผู้ใช้จะต้องเขียนโปรแกรมเป็นภาษาเครื่อง ซึ่งอยู่ในรูปของเลขฐานสอง เก็บไว้ในหน่วยความจำข้อมูล แต่ละคำสั่งของ 8031 อาจประกอบด้วย 1 2 หรือ 3 ไบต์ แล้วแต่ว่าจะเป็นคำสั่งให้ทำงานอะไร คอมพิวเตอร์ก็เหมือนกับคนที่ต้องทำงานตามคำสั่ง เมื่อรับคำสั่งก็จะนำไปตามคำสั่งนั้นเสร็จสิ้น แล้วก็จะกลับมารับคำสั่งต่อไป

เมื่อเริ่มป้อนไฟเลี้ยงให้กับ 8031 จะเริ่มจากบล็อค โปรแกรมเคาน์เตอร์ ซึ่งเป็นวงจรนับ (Counter Circuit) ชนิดหนึ่งส่งค่าตำแหน่งหน่วยความจำสำหรับโปรแกรมลงบัส หมายเลข 1 บัสมีขนาด 16 บิต ค่าตำแหน่งหน่วยความจำนี้ถูกส่งไปเก็บไว้ที่ Program ADDR Register ที่เป็นวงจรแลทช์ ข้อมูลซึ่งเป็นค่าตำแหน่งหน่วยความจำ จะปรากฏที่บนบัส 16 บิต หมายเลข 2 ถ้าเป็นค่าตำแหน่งหน่วยความจำแรกหลังจากรีเซต ค่าตำแหน่งหน่วยความจำจะเป็น 0000H หน่วยความจำสำหรับโปรแกรมจะเลือกได้ว่าเป็นรอม ภายในหรือภายนอก 8031 โดยการป้อนสภาวะลอจิกเข้าไปที่ 8031 ทางขา EA ซึ่งต่ออยู่กับส่วน Timing and Control ทำหน้าที่เป็นวงจรถอดรหัส (Decoder) แล้วสร้างสัญญาณควบคุมต่อไป ถ้าป้อนสัญญาณลอจิกที่ 0 เข้าไปที่ขา EA จะเป็นการเลือกใช้รอม ภายใน 8031 โดยที่วงจร Timing and Control จะสร้างสัญญาณไปยังรอมภายใน ให้ส่วนของข้อมูลที่เป็นคำสั่งจากตำแหน่งที่ถูกชี้ตัวค่าตำแหน่งที่ส่งมาทางบัสหมายเลข 2 ข้อมูลจากรอม จะถูกส่งไปยังบัสหมายเลข 3 ที่เรียกว่า Internal Data Bus แล้วนำไปเก็บไว้ที่ Instruction Register (เป็นวงจร Latch) เพื่อส่งต่อให้กับวงจร Timing and Control ทำการถอดรหัสและควบคุมการทำงานส่วนอื่น ๆต่อไปแล้วแต่ว่าเป็นคำสั่งที่ให้ทำงานอะไร ในกรณีที่เลือกรอม ภายนอก 8031 โดยป้อนสัญญาณลอจิก 1 เข้าไปที่ขา EA จะทำให้วงจร Timing and Control ส่งสัญญาณไปยังพอร์ท 0 และพอร์ท 2 เพื่อส่งค่าตำแหน่งหน่วยความจำบนบัสหมายเลข 2 ออกไปชี้หน่วยความจำภายนอกจากนั้นจะอ่านข้อมูลที่เป็นคำสั่งกลับเข้ามาทางพอร์ท 0 ไปยัง Internal Data Bus แล้วไปเก็บที่ Instruction Register เพื่อทำงานต่อไป เหมือนกับตอนอ่านคำสั่งจาก รอม ภายใน การทำงานในช่วงส่งค่าตำแหน่งหน่วยความจำแล้วอ่านข้อมูลที่เป็นคำสั่งกลับเข้ามาเก็บไว้ใน Instruction Register เรียกว่าเป็นช่วงของการ Fetch ช่วงต่อไปเป็น

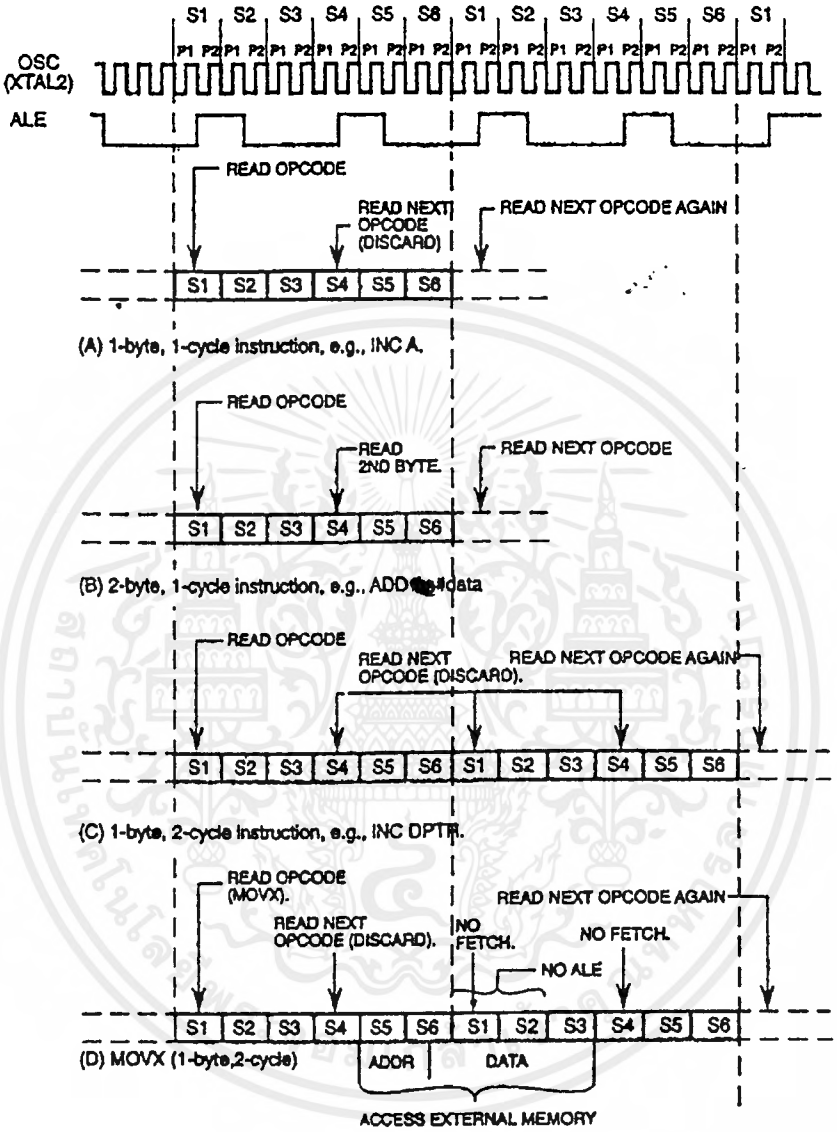
ช่วงของการทำงานตามคำสั่งเรียกว่า Execute เช่นถ้าเป็นคำสั่งให้บวกข้อมูลในรีจิสเตอร์แอคคิวมูเลเตอร์ กับข้อมูลจากหน่วยความจำข้อมูลภายใน แรมตำแหน่ง 23H วงจร Timing and Control ก็จะส่งสัญญาณให้ Instruction Register ส่งค่าตำแหน่งหน่วยความจำ 23H ไปยัง Internal Data Bus แล้วนำข้อมูลไปเก็บไว้ที่ RAM ADDR Register เพื่อชี้ตำแหน่งหน่วยความจำแรม จากนั้น Timing and Control จะสั่งให้แรม ส่งข้อมูลที่เก็บอยู่ภายในหน่วยความจำตำแหน่ง 23H ลงมายัง Internal Data Bus แล้วนำข้อมูลไปเก็บไว้ที่ TMP1 (วงจร Latch) ขณะเดียวกัน Timing and Control ก็จะส่งสัญญาณไปยัง ACC ให้ส่งข้อมูลมายัง TMP2 (วงจร Latch) วงจร ALU ซึ่งโครงสร้างเป็นวงจรคำนวณทางคณิตศาสตร์ (บวก , ลบ , คูณ ,หาร)และยังสามารถทำงานทางลอจิก (AND , OR , NOT , XOR) จะทำการบวกจาก TMP1 และ TMP2 เข้าด้วยกันผลลัพธ์ที่ได้จะส่งผ่าน Internal Data Bus กลับไปยัง ACC PSW (Program Status Word) ซึ่งจะทำหน้าที่เก็บสถานะผลลัพธ์ของการทำงานใน ALU เช่นผลลัพธ์การบวกมีค่าเกิน 8 บิต ก็จะทำให้บิตหนึ่งใน PSW ถูกเซตเป็น 1

การทำงานที่กล่าวมาข้างต้นจะขึ้นกับสัญญาณควบคุมที่สร้างมาจากวงจร Timing and Control และสัญญาณที่สร้างขึ้นนี้จะอ้างอิงกับสัญญาณนาฬิกาที่สร้างมาจากวงจรออสซิเลเตอร์ ทำให้การทำงานต่างๆ เป็นไปตามลำดับที่ผู้ผลิตได้ออกแบบไว้ ดังในรูป 2.18

คำสั่งแต่ละคำสั่งของ 8031 จะใช้เวลาทำงาน 1,2 หรือ 3 ไซเคิลของเครื่อง (Machine Cycle) แล้วแต่ว่าเป็นคำสั่งประเภทใด 1 ไซเคิลของเครื่องจะใช้เวลาประมาณ 12 ไซเคิลของสัญญาณนาฬิกา ดังนั้นแต่ละคำสั่งของ 8031 จะใช้เวลาการทำงาน 12 24 หรือ 36 ไซเคิลของสัญญาณนาฬิกานั่นเอง แต่ละไซเคิลของเครื่องจะถูกแบ่งเป็น 6 สถานะ คือ S1 , S2 , S3 , S4 , S5 และ S6 แต่ละสถานะ จะประกอบด้วย 2ไซเคิลของสัญญาณนาฬิกาในไซเคิลแรกจะเรียกว่า เฟส 1 และไซเคิลที่ 2 จะเรียกว่า เฟส 2 ในแต่ละเฟสจะนับตั้งแต่ขอบขาลงของสัญญาณนาฬิกาที่อยู่ถัดไปดังในรูปที่ 2.14 เมื่อ 8031 ทำงานเสร็จ 1 ไซเคิลของเครื่องก็จะเริ่มทำงาน สถานะ 1 เฟส 1 ของไซเคิลต่อไป ใน 1 ไซเคิลของเครื่องวงจร Timing and Control สร้างสัญญาณ ALE ออกมา 2 ไซเคิลเพื่อ Fetch เข้าไป 2 คำสั่งเสมอ ที่บริเวณขอบขาขึ้นของสัญญาณ ALE คำสั่งใด จะมีกี่ไบต์ หรือใช้เวลาทำงานกี่ไซเคิลจะดูได้จากตารางชุดคำสั่งของ 8031

คำสั่งประเภท 1 ไบต์ 1 ไซเคิล ของเครื่องได้แก่คำสั่ง INC A จะมีการอ่านคำสั่งจากหน่วยความจำสำหรับโปรแกรม 2 ครั้ง ที่เวลาประมาณขอบขาขึ้นของสัญญาณ ALE เมื่อคำสั่งแรกถูก อ่านเข้าไปที่เวลาขอบขาขึ้น ของสัญญาณ ALE แรก แล้วนำไปเก็บที่ Instruction Register เพื่อให้วงจร Timing and Control ถอดรหัสแล้วเข้าอยู่การ Execute ขณะเดียวกันก็เริ่มต้นการ Fetch คำสั่งที่อยู่ในหน่วยความจำตำแหน่งถัดไปเข้ามาและคำสั่งที่ 2 จะถูกอ่านเข้ามาที่ขอบเวลาขาขึ้นของสัญญาณ ALE ถัดไป วงจร Timing and Control เมื่อถอดรหัสคำสั่งแรกก็จะทราบว่าการทำงานคำสั่งนี้ให้สิ้นสุดจะ

ใช้คำสั่งเพียง 1 ไบต์ ดังนั้นคำสั่งที่ถูกอ่านมาไบต์ที่ 2 จะไม่ถูกนำมาทำงาน เพียงแต่อ่านเข้ามาแล้วทิ้งไป ดังในรูป 2.18



รูปที่ 2.18 ผังการทำงานของแต่ละคำสั่งใน 8031

คำสั่งประเภท 2 ไบต์ และใช้เวลา 1 ไซเคิลของเครื่องได้แก่ ADD A,#data ในหนึ่งไซเคิลของเครื่องนี้จะมีการอ่านเข้ามา 2 ไบต์ เหมือนคำสั่งประเภท 1 ไบต์ 1 ไซเคิลของเครื่อง แตกต่างกันที่ไบต์ที่ 2 จะถูกนำมาใช้งานด้วยไม่ถูกทิ้งไปดังในรูปที่ 2.14 b ตัวอย่างคำสั่ง ADD A,#33H จะเขียนเป็น

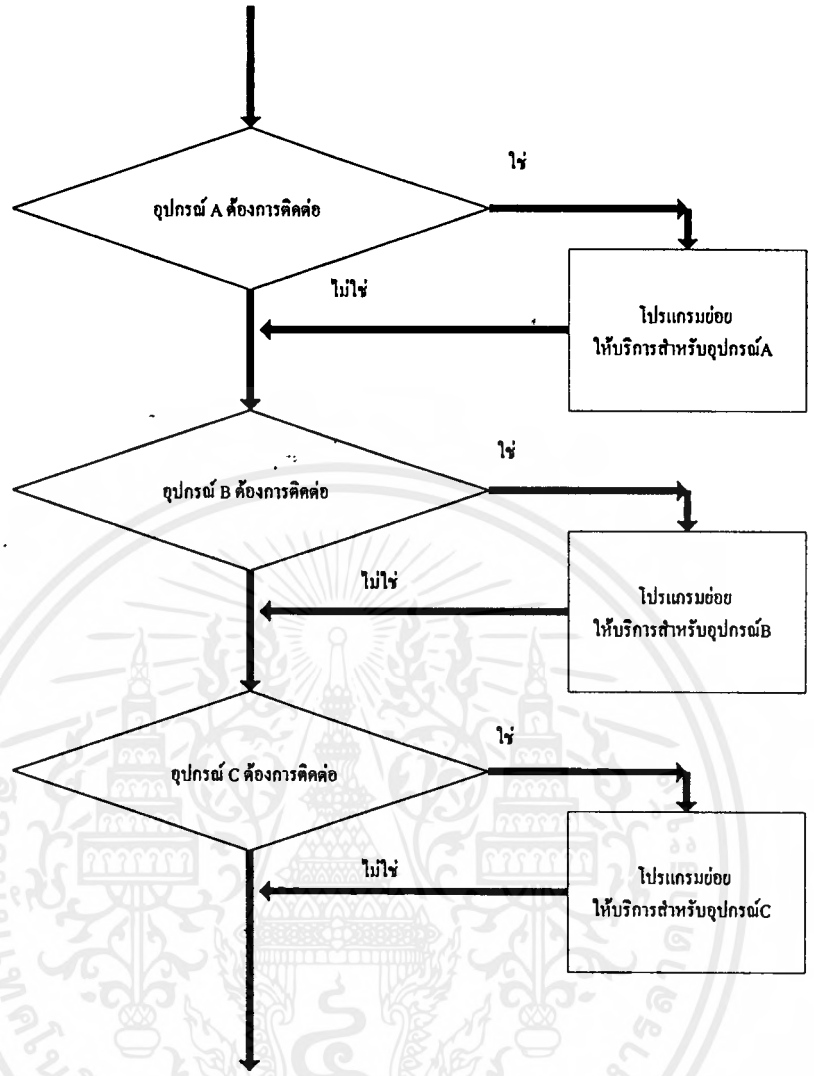
ภาษาเครื่องได้ 2 ไบต์ คือ 24 33 เมื่ออ่านคำสั่งแรกคือ 24 เข้าไปที่ Instruction Register แล้ว Timing and Control จะถอดรหัสพบว่าเป็นคำสั่งบวกเลข ก็จะส่งสัญญาณไปยังแอคคิวมูเลเตอร์ ให้เอาข้อมูลไปไว้ที่ TMP 1 เมื่อคำสั่งที่ 2 ถูกอ่านเข้ามาที่ Instruction Register แล้ว Timing and Control จะสั่งให้เอาข้อมูลไบต์ที่ 2 ส่งลงไปยัง Internal Data Bus ไปเก็บยัง TMP 2 จากนั้นวงจร ALU จะนำเอาข้อมูล TMP 1 และ TMP 2 มาบวกกันผลลัพธ์ที่ส่งออกจาก ALU ไปยัง Internal Data Bus แล้วนำไปเก็บไว้ที่ แอคคิวมูเลเตอร์

คำสั่งประเภท 1 2 หรือ 3 ไบต์ ที่ใช้เวลาทำงาน 2 ไซเคิล ของเครื่องเช่น คำสั่ง INC DPTR จะมีการอ่านคำสั่งเข้าไป 4 ครั้งทุก ๆ ขอบขาขึ้นของสัญญาณที่มี ALE ที่มี 2ครั้ง ต่อ 1 ไซเคิล ของเครื่อง ถ้าเป็นคำสั่งประเภท 1 2 หรือ 3 ไบต์ วงจร Timing and Control จะเอาคำสั่ง 1 2 หรือ 3 ไบต์แรกเท่านั้นไปทำงานส่วนคำสั่งที่เหลือทิ้งไปดังในรูปที่ 2.19 C คำสั่ง 1 ไบต์ ที่ใช้เวลาทำงาน 2 ไซเคิล ของเครื่องที่กล่าวมาแล้วจะไม่รวมถึงคำสั่งMOVX ซึ่งใช้ในการอ่านหรือเขียนข้อมูลกับหน่วยความจำหน่วยความจำข้อมูลภายนอก การทำงานของคำสั่งนี้จะมีการ Fetch คำสั่งเข้าไป 2 ไบต์ ในไซเคิลของเครื่องแรก ในไซเคิลของเครื่องที่2 จะไม่มีการ Fetch คำสั่งเข้าไปแต่จะเป็นช่วงเวลาของการอ่านหรือการเขียนข้อมูลกับหน่วยความจำข้อมูลภายนอก สัญญาณ ALE ซึ่งปกติจะเปลี่ยนเป็น 1 ที่ S1 P2 ก็จะไม่เปลี่ยนเป็น 1 ในไซเคิลของเครื่องที่ 2 โดยจะเป็น 0 จนกว่าจะถึงเวลา S4 P2 ของไซเคิลของเครื่องที่ 2 สัญญาณ ALU จะเปลี่ยนเป็น 1 เพื่อทำการอ่านหรือเขียนข้อมูลกับ หน่วยความจำข้อมูลภายนอก

2.8 การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ 8031

การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ในงานต่าง ๆ ส่วนมากมักจะเป็นแบบถ่ายข้อมูลระหว่างวงจรภายนอกกับไมโครคอนโทรลเลอร์ หรือหน่วยการทำงานในตัวไมโครคอนโทรลเลอร์เอง ซึ่งการเคลื่อนย้ายหรือแลกเปลี่ยนข้อมูลลักษณะเช่นนี้รวมถึงการเคลื่อนย้ายข้อมูล ระหว่างในไมโครคอนโทรลเลอร์กับหน่วยความจำมักจะเรียกโดยรวมว่า การอินพุท เอาท์พุท โดยหลักการแล้วมีเทคนิคที่ใช้กันอยู่ 3 วิธีคือ การโพลลิ่ง (Polling) การอินเตอร์รัปต์ (Interrupt) และดีเอ็มเอ (DMA)

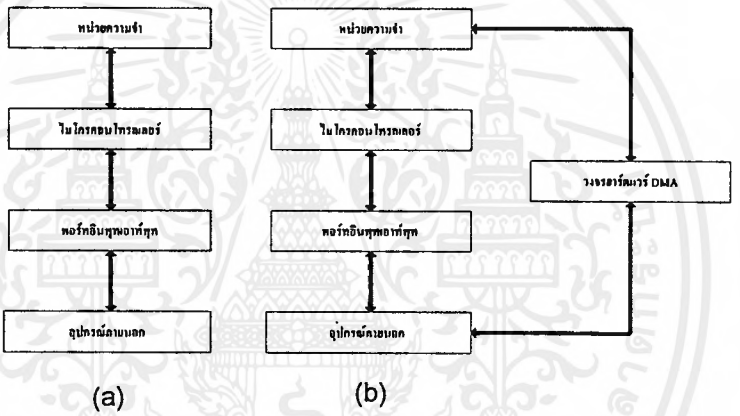
หากพิจารณาในลักษณะโปรแกรมควบคุมของไมโครคอนโทรลเลอร์แล้ว หลักการทั้งสามนี้จะแตกต่างกันออกไปโดย วิธีการโพลลิ่ง (Polling) หรือมักจะเรียกอีกชื่อหนึ่งว่า Program Controlled I/O เป็นการอินพุท/เอาท์พุท ภายใต้ความควบคุมจากโปรแกรมทั้งสิ้นกล่าวคือเมื่อใดก็ตามที่มีการใช้ชุดคำสั่งสำหรับการรับหรือส่งข้อมูล จึงจะมีการอินพุทและเอาท์พุทเกิดขึ้นตามลำดับเท่านั้น โดยในบางครั้งอาจจะต้องมีการตรวจสอบสถานะความพร้อมของอุปกรณ์ ก่อนที่การรับ/ส่ง ข้อมูลเกิดขึ้น ซึ่งแสดงว่าพร้อมจึงจะทำการเริ่มต้นการส่งข้อมูลระหว่างกัน กระบวนการนี้มักจะเรียกว่า Hand Shaking ดังแสดงเป็นแผนภาพในรูปที่ 2.19



รูปที่ 2.19 ลำดับการทำงานแบบโพลลิ่ง

สำหรับเทคนิควิธีการ อินเตอร์รัปต์ (Intrerupt) จะเป็นในลักษณะที่ตรงกันข้ามกับเทคนิควิธีที่ผ่านมา โดยอุปกรณ์ภายนอกจะบอกถึงสภาวะความพร้อมโดยการปรับสภาวะของขาสัญญาณที่สนับสนุนวิธีการอินเตอร์รัปต์นี้ให้เหมาะสม เพื่อแจ้งไมโครคอนโทรลเลอร์ทราบโดยตรงว่าต้องการที่จะโอนย้ายข้อมูลระหว่างกันเมื่อไมโครคอนโทรลเลอร์ถูกอินเตอร์รัปต์หรือเรียกว่า ขัดจังหวะ ก็จะหยุดการทำงานของโปรแกรมชั่วขณะหนึ่งและเกิดการควบคุมเพื่อสั่งงานให้ไปประมวลผลยังส่วนของโปรแกรมย่อยที่ตอบสนองการอินเตอร์รัปต์ ดังนั้นการทำงานตามลักษณะเทคนิควิธีการอินเตอร์รัปต์ การโอนถ่ายข้อมูลจะเริ่มต้นจากการอุปกรณ์ฮาร์ดแวร์ภายนอก และต่อมาจึงถูกควบคุมการทำงานจากโปรแกรมย่อยตอบสนองการอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์

ส่วนเทคนิคดีเอ็มเอนั้น เป็นการควบคุมการโอนถ่ายข้อมูล โดยใช้วงจรฮาร์ดแวร์ทำหน้าที่ดำเนินการ โดยคำว่า ดีเอ็มเอ (DMA) นั้นย่อมาจาก Direct Memory Access หมายถึงการโอนถ่ายข้อมูลระหว่างอุปกรณ์ อินพุท/เอาท์พุท ภายนอกกับหน่วยความจำ โดยไม่ต้องผ่านรีจิสเตอร์ของไมโครคอนโทรลเลอร์เลย อย่างไรก็ตามหน้าที่เริ่มต้นการติดต่อก็ยังเป็นของไมโครคอนโทรลเลอร์แต่เป็นเพียงการบอกข้อมูลที่เกี่ยวกับ หน่วยความจำ กระบวนการที่เกิดขึ้นจริงๆ นั้นเป็นหน้าที่ของวงจรฮาร์ดแวร์ ที่ควบคุมการทำงานแบบดีเอ็มเอ นั่นเอง ขอให้ดูแผนภาพการทำงานเปรียบเทียบในรูปที่ 2.20 เทคนิคการทำงานแบบนี้มักจะนำมาใช้กับการโอนถ่ายข้อมูลจำนวนมาก ๆ และต้องการความรวดเร็วอย่างมากด้วย อย่างไรก็ตามเรามักจะพบอุปกรณ์ฮาร์ดแวร์ DMA ที่ใช้ในไมโครคอนโทรลเลอร์ สำหรับควบคุมน้อยมาก



รูปที่ 2.20 แผนภาพการทำงานอย่างง่ายของการติดต่อระหว่างอุปกรณ์ภายนอกกับหน่วยความจำ
(a) โดยผ่านทางไมโครคอนโทรลเลอร์
(b) โดยผ่านทางวงจรฮาร์ดแวร์

2.9 คำสั่งการใช้งานพอร์ทอินพุท/เอาท์พุท

เมื่อพิจารณาถึงวิธีการส่งข้อมูลภายในพอร์ท จะสามารถแยกประเภทพอร์ทออกได้เป็นพอร์ทแบบขนาน (Parallel Port) ซึ่งทำการส่งจำนวนบิตข้อมูลทั้งหมดออกมา หรือนำเข้าไปพร้อมกันในคราวเดียวกัน และพอร์ทแบบอนุกรม(Serial Port) ซึ่งทำการโอนย้ายข้อมูลคราวละบิตจนครบ ใช้หลักการที่เรียกว่า Memory Mapped System กล่าวคือ การอ้างถึงพอร์ท รีจิสเตอร์หรืออุปกรณ์ต่าง ๆ ภายในระบบ จะเป็นการติดต่อกับหน่วยความจำ ตำแหน่งหนึ่งเท่านั้น ดังนั้นในการดำเนินการเพื่อนำเข้าหรือส่งข้อมูลกับพอร์ท จึงใช้คำสั่งการอ่านค่าจากหน่วยความจำ ซึ่งถูกแบ่งออกแบบให้เป็นตำแหน่ง

ของพอร์ท หรือคำสั่งการเขียนค่าข้อมูล ไปยังหน่วยความจำนั้นแทน ดังนั้นจะสังเกตได้ว่าในตารางชุดคำสั่งของ 8031 จะไม่มีคำสั่งที่เกี่ยวกับการทำงานของพอร์ทแต่ประการใด เช่น คำสั่ง IN (นำเข้าข้อมูลจากพอร์ท) หรือคำสั่ง (ส่งข้อมูลออกทางพอร์ท) เป็นต้น นอกจากนี้ยังสามารถใช้คำสั่งจัดการข้อมูลแบบบิตได้โดยตรง (Single-bit Operation) เพื่อมาจัดการพอร์ทอินพุท/เอาท์พุททั้งหมดแบบเส้นสัญญาณเดี่ยวได้ โดยการใช้คำสั่ง SETB เพื่อกำหนดค่าเป็น 1 หรือคำสั่ง CLR เพื่อทำให้บิตมีค่าเป็น 0 คำสั่งเหล่านี้มีประโยชน์และทำให้ลดความซับซ้อนในการใช้คำสั่งภายในโปรแกรมลงได้มาก การพิจารณาตรวจสอบค่าบิตของแต่ละพอร์ท อินพุท/เอาท์พุทนั้นก็สามารถใช้ชุดคำสั่ง ในการทดสอบบิตของพอร์ทได้โดยตรงโดยไม่ต้องมีการย้ายค่าของพอร์ทไปยังรีจิสเตอร์ก่อนแต่ประการใด

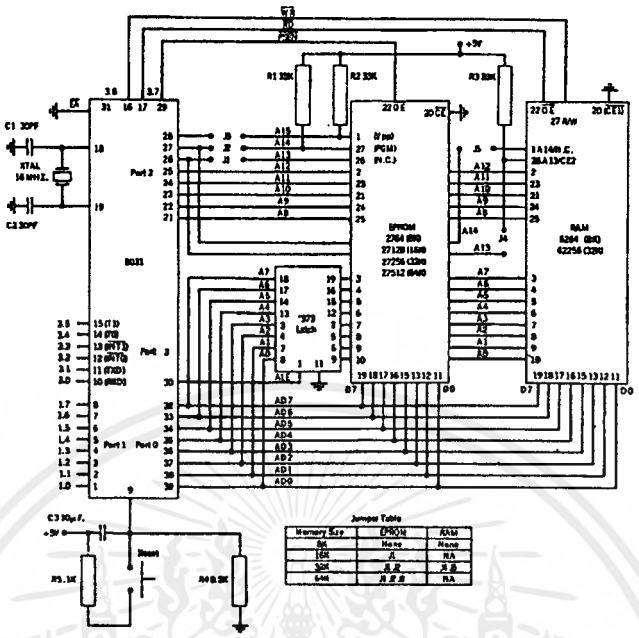
2.9.1 พอร์ทแบบขนานของ 8031

8031 มีโครงสร้างใช้งานแบบขนานได้จำนวนทั้งหมด 4 พอร์ท เรียกชื่อเรียงตามลำดับว่า พอร์ท 0 1 2 และ 3 และเป็นพอร์ทขนานแปดบิตทั้งหมด การใช้งานพอร์ทสามารถทำได้ทั้งในลักษณะของเส้นสัญญาณเดี่ยว ๆ หรือกลุ่มของสัญญาณได้ นอกจากนี้พอร์ท 0 2 และ 3 ยังสามารถนำไปใช้งานอื่น ๆ ที่ไม่ใช่เป็นพอร์ทอินพุท/เอาท์พุทได้ โดยพอร์ท 0 จะทำหน้าที่เป็นมัลติเพล็กซ์ระหว่างบัสแอดเดรสไบต์ต่ำและบัสข้อมูลสำหรับการติดต่อกับวงจรประกอบร่วมกับข้อมูลบัสแอดเดรสไบต์สูงซึ่งจะส่งออกมาทางพอร์ท 2 สำหรับพอร์ท 3 นั้นนอกเหนือไปจากความสามารถเช่นพอร์ทปกติแล้ว สามารถนำไปเป็นขาสัญญาณของการอินเตอร์รัปต์ต่าง ๆ ซึ่งรวมทั้งสร้างสัญญาณควบคุม /RD และ/WR เพื่อทำหน้าที่อ่านหรือเขียนหน่วยความจำข้อมูลภายนอกด้วย การใช้งานพอร์ทลักษณะงานแบบอื่น ๆ ที่ไม่ใช่ เป็น พอร์ทแบบอินพุท/เอาท์พุท นี้จะดำเนินการโดย 8031 เองโดยอัตโนมัติ

2.10 ตัวอย่างวงจรการต่อพ่วงกับอุปกรณ์ภายนอก

2.10.1 วงจรการใช้งานร่วมกับอีพรอมและแรมภายนอก

การพัฒนาโปรแกรมเพื่อให้เครื่องต้นแบบที่ใช้ไมโครคอนโทรลเลอร์ ทำงานได้ตามข้อกำหนดต่าง ๆ อย่างไม่ผิดพลาด ส่วนใหญ่จะพัฒนาโปรแกรมด้วยการให้ตัวไมโครคอนโทรลเลอร์ MCS-51 ตัวนี้ทำงานตามโปรแกรมที่เก็บอยู่ภายนอก ซึ่งจะใช้ตัวอีพรอม แต่ระหว่างการพัฒนาอาจจะใช้หน่วยความจำข้อมูลที่มีแบตเตอรี่จ่ายไฟสำรอง หรืออีพรอมอีมูเลเตอร์ ที่รับส่งข้อมูลอนุกรมกับพีซีก็ได้ เพื่อเพิ่มความสะดวกในการอ่านและเขียนแก้ไขโปรแกรม ลักษณะการต่อวงจรที่ให้ทำงานตามโปรแกรมที่อยู่ภายนอก แสดงในรูปที่2.21

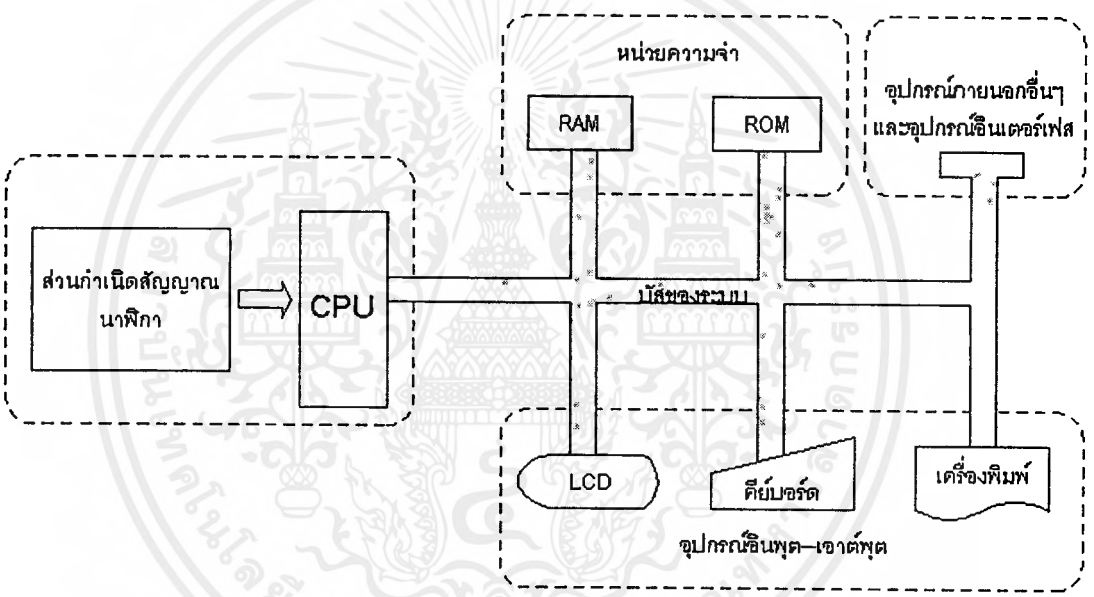


รูปที่ 2.21 วงจรไมโครคอนโทรลเลอร์ที่ใช้อีพรอมภายนอก

โดยที่วงจรจะบังคับให้ขา EA มีสถานะต่ำ เพื่อเป็นการบอกให้ไมโครคอนโทรลเลอร์ทำงานโปรแกรมภายนอก 74LS373 จะทำการแลทช์ข้อมูลแอดเดรสไว้ เนื่องจากขา DB0-DB7 เป็นแบบมัลติเพล็กซ์ข้อมูลและแอดเดรส ดังนั้นจึงต้องแลทช์แยกเอาค่าแอดเดรส มาแสดงที่ขาเอาท์พุทของ 74LS373 ไปถอดรหัสโปรแกรมในตัวอีพรอม เพื่อให้ตัวไมโครคอนโทรลเลอร์สามารถทำงานตามโปรแกรมภายนอกได้ถูกต้อง ข้อสังเกต เมื่อเราใช้โปรแกรมจากภายนอก ไมโครคอมพิวเตอร์จะทำให้ตัวไมโครคอนโทรลเลอร์มีขาพอร์ทที่ลดน้อยไปประมาณ 16 ขา คือขา DB0-DB7 และขา P20-P27 ส่วนการเข้าถึงข้อมูลแรม จะใช้ขา RD,WR เป็นขาควบคุม

บทที่ 3
การออกแบบโครงสร้างทางฮาร์ดแวร์

โครงสร้างโดยทั่วไปของซิงเกิลบอร์ด นั้นมีลักษณะคล้ายกับไมโครคอมพิวเตอร์ ซึ่งนอกจากจะประกอบด้วยตัวไมโครคอนโทรลเลอร์แล้ว ยังต้องประกอบไปด้วย หน่วยความจำ (หน่วยความจำโปรแกรมซึ่งเก็บไว้ในรอม และหน่วยความจำข้อมูล ซึ่งเก็บไว้ในแรม) อุปกรณ์อินพุท เอาท์พุท ได้แก่ หน่วยแสดงผล หน่วยรับข้อมูล ซึ่งทั้งหมดนี้จะทำงานร่วมกับไมโครคอนโทรลเลอร์ ดังรูปที่ 3.1

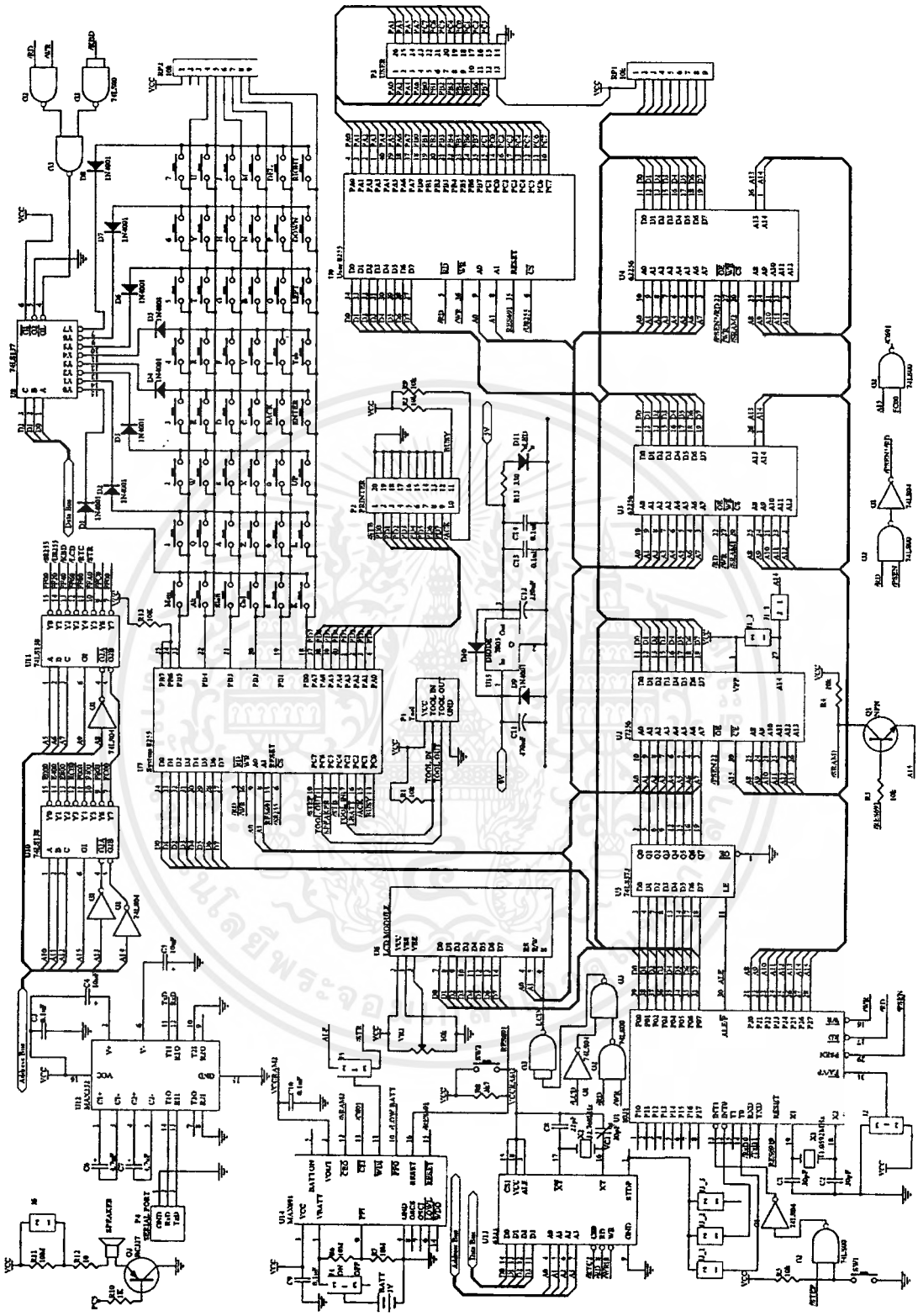


รูปที่ 3.1 ส่วนประกอบของซิงเกิลบอร์ด

จากโครงสร้างดังรูปที่ 3.1 สามารถออกแบบวงจรรวมทั้งหมด ได้ดังรูปที่ 3.2 ซึ่งการทำงานของส่วนต่าง ๆ เป็นดังนี้

3.1 วงจรควบคุมหลัก

วงจรควบคุมหลัก แสดงดังรูปที่ 3.2 ซึ่งส่วนที่สำคัญที่สุดของวงจรในส่วนนี้ คือ ไมโครคอนโทรลเลอร์ เบอร์ 8031 ที่พอร์ท 0 ซึ่งเป็น แอดเดรสบัส และดาต้าบัส ต่อกับ 74LS373 (8 bits - Latch) เพื่อทำการดีมัลติเพล็กซ์ (Demultiplex) แอดเดรสไปที่ (A0-A7) ออกจากดาต้า ซึ่งการควบคุมการดีมัลติเพล็กซ์จะใช้ สัญญาณ ALE (Address Latch Enable) เป็นตัวควบคุม ขา EA/VP เป็นขาที่ใช้



รูปที่ 3.2 วงจรรวมของซิงเกิลบอร์ด

ในการเลือกให้ไมโครคอนโทรลเลอร์ อ่านคำสั่งจากหน่วยความจำโปรแกรม จากภายในหรือภายนอก โดยการเลือกจัมเปอร์ J2 ซึ่งปกติจะต่อลงกราวนด์เพื่อที่จะใช้หน่วยความจำโปรแกรมภายนอก ถ้าหากต้องการให้ไมโครคอนโทรลเลอร์อ่านคำสั่งจาก ROM ภายใน ก็ทำได้โดยการ เซ็ท J2 ให้ลงกราวนด์

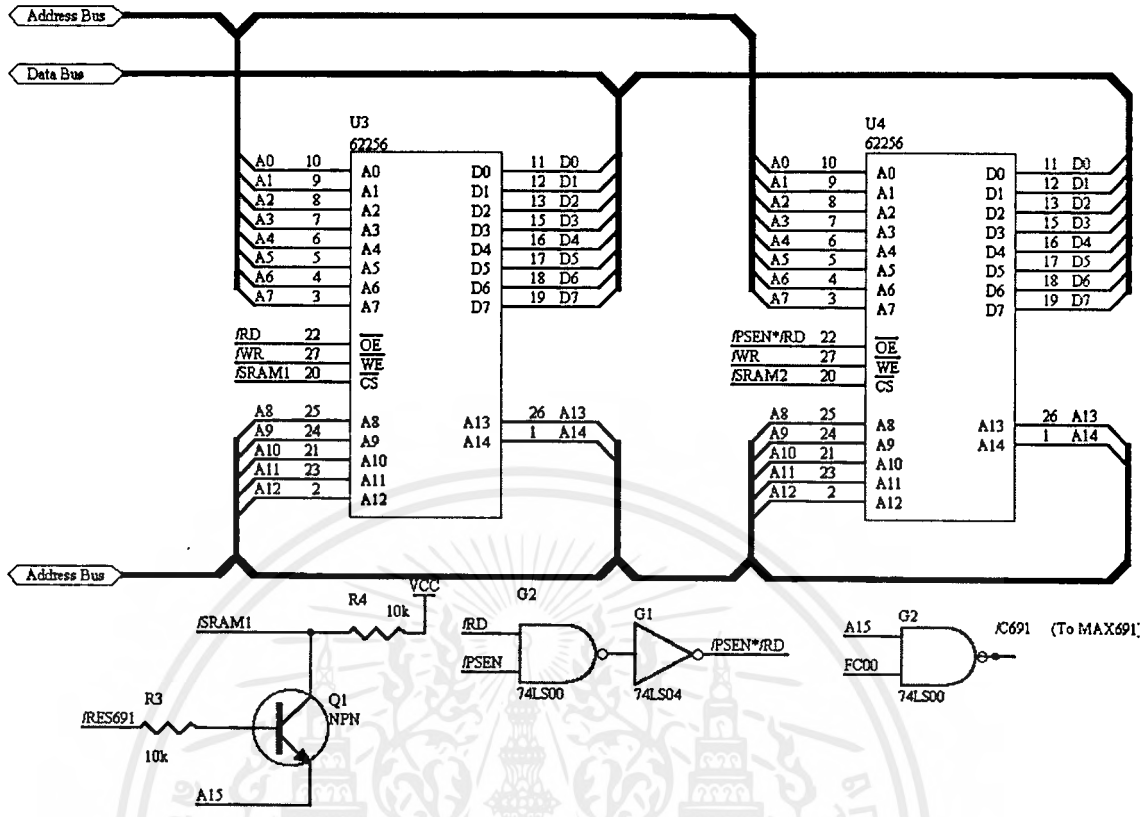
ขา X1และ X2 ต่ออยู่กับคริสตอลซึ่งจะใช้คริสตอลที่มีความถี่ 11.0592 MHz ซึ่งความถี่ค่านี้จะทำให้ไมโครคอนโทรลเลอร์ สามารถกำหนดคาบเวลาในการสื่อสารแบบอะซิงโครนัส ผ่านทางขา TXD และ RXD ได้ใกล้เคียงกับค่ามาตราฐานมากที่สุด ที่ขา INT1 ต่อกับสัญญาณที่ได้จากการ OR กันของสัญญาณ Single-Step ที่สร้างมาจาก System 8255 (U7) และสัญญาณ Break ที่ได้จากการกดสวิตช์ Break ซึ่งทำให้สามารถสร้างสัญญาณอินเตอร์รัปต์ได้สองทาง คือทางฮาร์ดแวร์ โดยสั่งผ่านทางสวิตช์ และทางซอฟท์แวร์ โดยสั่งผ่านทาง System 8255 นั่นเอง

3.2 การถอดรหัสตำแหน่งของหน่วยความจำทั่วไป

หน่วยความจำข้อมูลจะถูกแบ่งออกเป็น 2 ส่วน คือช่วงแอดเดรส 0000- 7FFFH ซึ่งหน่วยความจำในส่วนนี้จะเป็นหน่วยความจำของผู้ใช้ และอีกส่วนหนึ่ง คือช่วงแอดเดรส 8000-FC00H ซึ่งหน่วยความจำในช่วงนี้ จะเป็นหน่วยความจำข้อมูล และเป็นหน่วยความจำโปรแกรม ในกรณีที่ผู้ใช้ต้องการเขียนโปรแกรมขึ้นมาใช้งาน

-ส่วนที่1 เริ่มตั้งแต่แอดเดรสที่ 0000-7FFFH ในรูปที่ 3.3 คือ U3 ขา C5 ของ RAM ตัวนี้จะต่อกับสัญญาณ /SRAM1 ซึ่งสัญญาณนี้ ได้ผ่านวงจรป้องกันความผิดพลาดของข้อมูลเมื่อเกิดสภาวะไฟตก ซึ่งมีหลักการทำงานคือ ในสภาวะปกติ ขาสัญญาณ /RES691 จะมี ลอจิก 1 (จะกล่าวถึงที่มาของสัญญาณนี้ในส่วนต่อไป) ดังนั้นทรานซิสเตอร์ Q1 จะทำงานหรือไม่ก็ขึ้นกับ A15 ในกรณีที่ติดต่อกับแรมตัวนี้ ซึ่งมีแอดเดรสอยู่ที่ 0000-7FFFH จะเห็นว่า A15 จะเป็นลอจิก 0 ซึ่งจะทำให้สัญญาณ /SRAM 1 มีลอจิก 0 ด้วย จึงสามารถทำการอ่านหรือเขียนแรมตัวนี้ได้ตามต้องการ

ในกรณีที่ไฟตกขาสัญญาณ /RES691 จะมีลอจิก 0 ซึ่งทรานซิสเตอร์ Q1 จะไม่ทำงาน ดังนั้นสัญญาณ /SRAM1 จะมีลอจิก 1 ซึ่งจะทำให้ไม่สามารถทำการติดต่อกับแรมตัวนี้ได้ จึงเป็นการป้องกันการสูญหายหรือเปลี่ยนแปลงข้อมูลในแรมตัวดังกล่าว ขาไฟเลี้ยงของแรมตัวนี้ได้ทำการต่อสำรองไว้เช่นกัน ทำให้ข้อมูลที่อยู่ภายในแรมไม่สูญหาย ถึงแม้ไฟของระบบจะถูกตัดไป



รูปที่3.3 วงจรถอดรหัสหน่วยความจำ

-**ส่วนที่ 2** เริ่มจากแอดเดรส 8000H-FC00H ก็คือแรม U4 ดังรูป 3.3 เนื่องจากหน่วยความจำในส่วนนี้ต้องการให้เป็นหน่วยความจำข้อมูลและหน่วยความจำโปรแกรมสำหรับผู้ใช้ในตัวเดียวกัน ดังนั้นขา /OE ของแรมตัวนี้แทนที่จะต่อกับสัญญาณ /RD เพียงอย่างเดียว จึงต้องนำมา AND กับสัญญาณ /PSEN ก่อนเพื่อที่จะสามารถทำให้หน่วยความจำข้อมูลตัวนี้ สามารถเป็นหน่วยความจำโปรแกรมได้ด้วย

ขา /CS ของแรมในช่วงนี้ต่อกับสัญญาณ /SRAM2 ซึ่งมาจาก U14 (MAX691) ซึ่งจะเป็นการป้องกันสภาวะไฟตก ดังเช่นแรมตัวแรก (0000-7FFFH) แต่แรมตัวนี้จะให้ U14 เป็นตัวป้องกันให้ (ซึ่งจะกล่าวถึงรายละเอียด ในส่วนของการทำงานของ MAX691) มาพิจารณาสัญญาณ /C691 ซึ่งเป็นสัญญาณที่ได้จากการถอดรหัสแอดเดรสของหน่วยความจำในช่วงนี้ก่อนส่งไปให้ MAX691 ซึ่งขาสัญญาณ /C691 ได้จากการ NAND กันระหว่างสัญญาณ A15 กับ สัญญาณ FC00 (มาจาก 74LS138) ซึ่งสาเหตุที่ต้องนำ A15 มา NAND กับ FC00 ก่อนแทนที่จะส่ง A15 ไปที่ MAX691 โดยตรง ก็เพื่อแยกแอดเดรสของหน่วยความจำกับ I/O ออกจากกัน คือกรณีที่เราไม่ได้อ้างถึง I/O สัญญาณ A15 จะมีลอจิก 1และ FC00 จะมีลอจิก 1 ด้วย (จะกล่าวถึงรายละเอียดในส่วนของวงจรถอดรหัสI/O) ดังนั้น /C691 จะมีลอจิก 0 ทำให้ติอต่อกับแรมได้ตามต้องการ เมื่อมีการอ้างถึง I/O FC00 จะเป็น 0 ทันที ทำ

ให้ /C691 จะเป็นลอจิก 1 ซึ่งทำให้ไม่สามารถติดต่อกับแรมได้ ดังนั้นแอดเดรสของส่วนนี้กับ I/O จึงไม่ซ้ำกัน

3.3 การถอดรหัสตำแหน่งของ I/O

การถอดรหัสตำแน่งจะใช้ไอซี 74LS138 จำนวน 2 ตัว (U10 และ U11) ดังรูปที่ 3.4 ซึ่งจะได้แอดเดรสของสัญญาณ I/O ต่าง ๆ ดังนี้

/S8255 คือสัญญาณเลือก 8255 ตัวที่ใช้งานในระบบ (System 8255) จากรูปที่3.2 คือ U7 ซึ่ง 8255 ตัวนี้มีหน้าที่เกี่ยวกับส่วนต่าง ๆ ของระบบ เช่น อ่านสัญญาณจากการสแกนคีย์บอร์ด , ส่งสัญญาณให้กับลำโพง ,ติดต่อกับพอร์ทเครื่องมือ (Tool Port) , ติดต่อกับพอร์ทพรินเตอร์ ฯลฯ ซึ่งจะมีแอดเดรสเท่ากับ FF00H

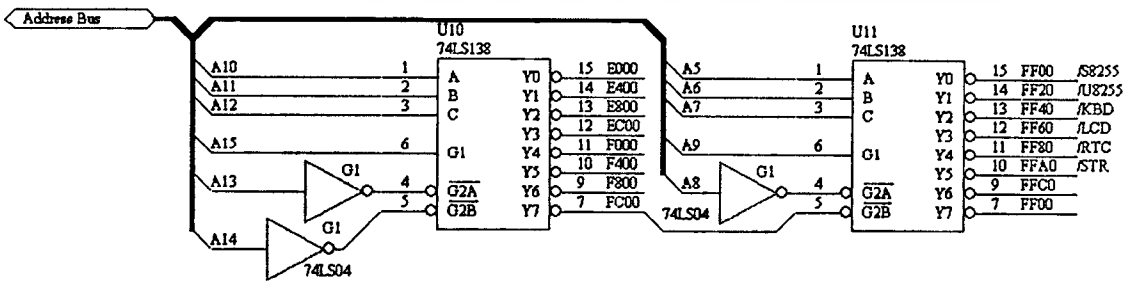
/U8255 คือสัญญาณเลือก 8255 ตัวที่ทำไว้ให้กับผู้ใช้ (User 8255) จากรูปที่ 3.2 คือ U9 ซึ่ง 8255 ตัวนี้จะทำให้ผู้ใช้โดยเฉพาะ ซึ่งผู้ใช้งานสามารถนำไปใช้ได้อย่างอิสระ มีแอดเดรสเท่ากับ FF20H

/KBD คือสัญญาณที่ใช้ในการสแกนคีย์บอร์ด ซึ่งสัญญาณนี้จะเป็นตัวไปเลือกให้ไอซี 74LS137 (U8) ทำงาน ซึ่งไอซีตัวนี้ทำหน้าที่ในการสแกนคีย์บอร์ดมีแอดเดรสเท่ากับ FF40H

/LCD คือสัญญาณในการเลือกติดต่อ กับ LCD (U6) ซึ่งเป็นหน่วยแสดงผลมีแอดเดรสเท่ากับ FF60H

/RTC คือสัญญาณเลือกติดต่อกับไอซี 6242 (U13) ซึ่งเป็นไอซี ที่ใช้สร้างฐานเวลาจริง (Real Time Clock) มีแอดเดรส เท่ากับ FF80H

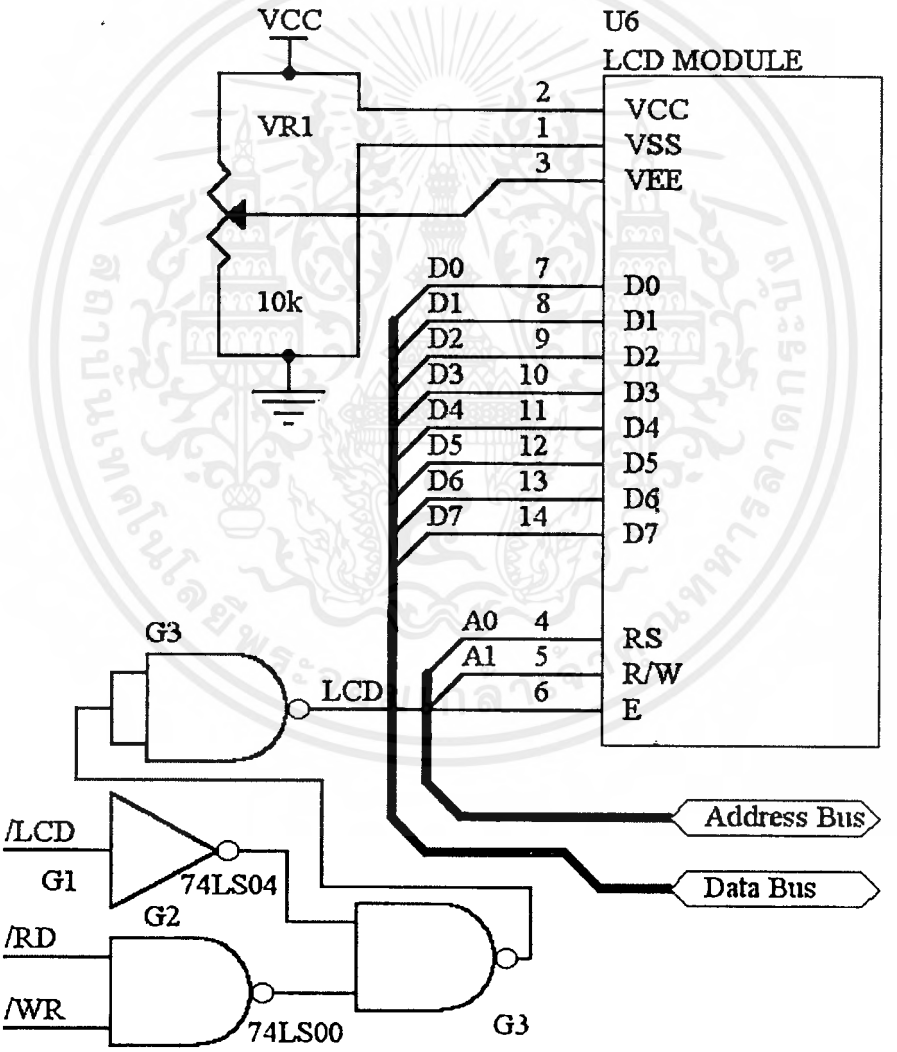
/STR เป็นขาสัญญาณที่ใช้ในการกระตุ้น (Strobe) ให้กับ วอทช์ดอก (Watch Dog) เพื่อให้วอทช์ดอก ทราบว่าขณะนี้ ระบบยังทำงานปกติ มีแอดเดรสเท่ากับ FFA0H



รูปที่ 3.4 วงจรถอดรหัสตำแหน่งของ I/O

3.4 หน่วยแสดงผล

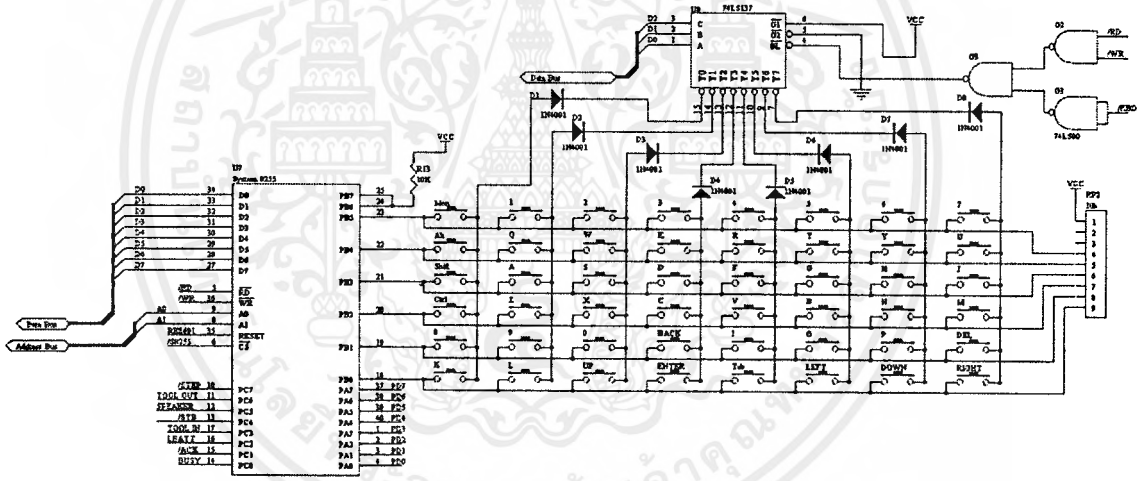
ในส่วนแสดงผลนี้จะใช้ LCD (Liquid Crystal Display) แบบ 4 แถว แถวละ 20 ตัวอักษร การเชื่อมต่อแสดงดังรูปที่ 3.5 สัญญาณที่นำมาเลือก LCD คือสัญญาณ /LCD ที่มีแอดเดรสอยู่ที่ FF60H แต่การติดต่อกับ LCD สัญญาณที่เข้ามาที่ขา E จะต้องเป็น ลอจิก 1 ดังนั้นสัญญาณ /LCD จึงต้องผ่านอินเวอร์เตอร์ แล้วนำไปทำการลอจิกแอนด์กับสัญญาณ /RD กับ /WD ก่อนจะนำมาต่อที่ขา E เพื่อทำให้เวลาที่ทำการอ่านหรือเขียนข้อมูลตรงกับเวลาที่ทำการเลือก LCD ขา RS และ R/W จะต่อกับ AO และ A1 ตามลำดับ VR1 มีไว้สำหรับปรับความเข้มของการแสดงผล



รูปที่ 3.5 วงจรของหน่วยแสดงผล

3.5 คีย์บอร์ด

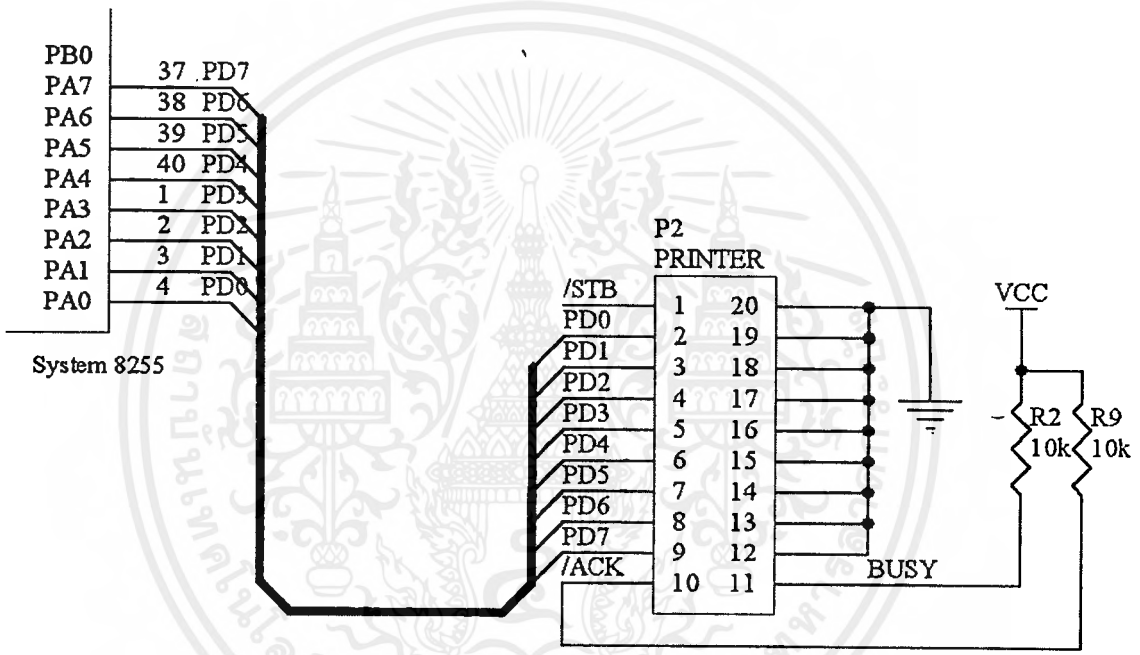
วงจรคีย์บอร์ดแสดงดังรูปที่ 3.6 เมื่อต้องการทำการสแกนจะต้องส่งสัญญาณ /KBD (FF40H) ให้ไปทำการลอจิกแอนด์กับสัญญาณ /RD และ /WR ก่อนเช่นเดียวกับส่วนของ LCD แล้วจึงส่งให้กับไอซี 74LS137 (U8) ซึ่งเป็นไอซี 3 to 8 Decoder/Latch จากนั้นก็ส่งสัญญาณผ่านทางคอลัมน์ที่ต้องการสแกน การสแกนจะกระทำทีละคอลัมน์ จากนั้นทำการตรวจสอบค่าคีย์ที่ถูกกดทาง พอร์ท B ของ System 8255 (U7) ซึ่งถ้าหากไม่มีคีย์ใดถูกกด ค่าที่อ่านได้จะเป็น 0FFH



รูปที่ 3.6 วงจรสแกนคีย์บอร์ด

3.6 การติดต่อกับพอร์ทพรินเตอร์

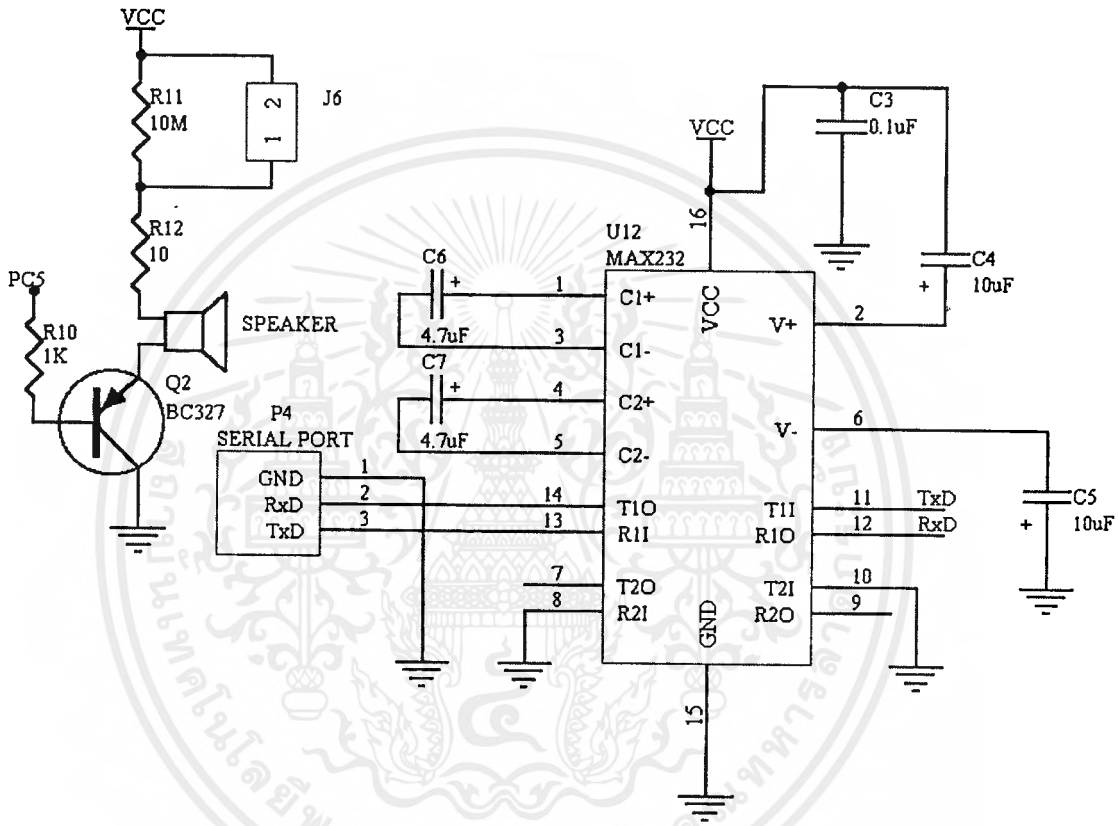
การติดต่อกับพอร์ทพรินเตอร์ จะติดต่อผ่านทางพอร์ท A ของ System 8255 โดย การส่งสัญญาณสโตรปให้กับพรินเตอร์ผ่านทาง PC4 และอ่านสถานะความพร้อมของพรินเตอร์กลับมาทาง PC0 ของ 8255 ตัวเดียวกัน ดังรูปที่ 3.7



รูปที่ 3.7 การติดต่อกับพอร์ทพรินเตอร์

3.7 วงจรสื่อสารผ่านพอร์ทอนุกรม

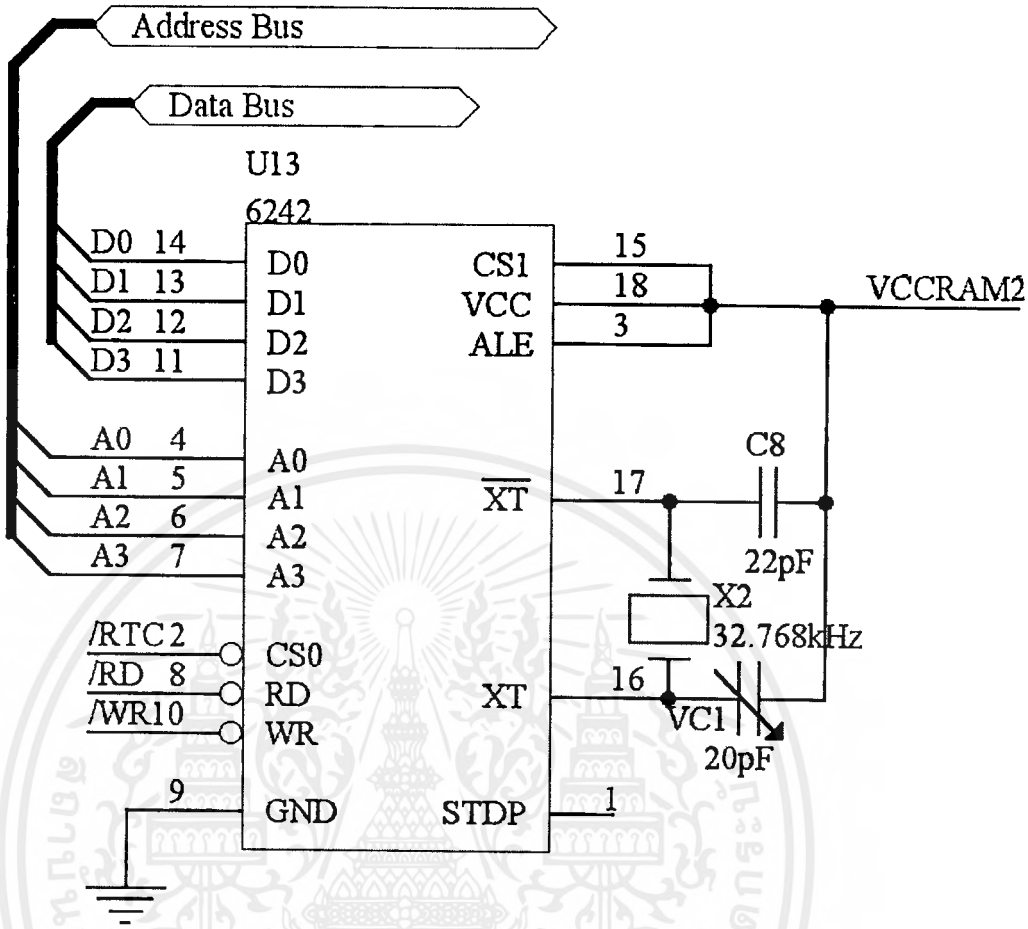
การสื่อสารผ่านพอร์ทอนุกรมจะใช้การเชื่อมต่อตามมาตรฐาน RS-232 เพื่อให้มีการใช้งานสอดคล้องกับอุปกรณ์คอมพิวเตอร์ต่าง ๆ โดยใช้ไอซีเบอร์ MAX232 (U12) ซึ่งมีการเชื่อมต่อดังรูปที่ 3.8 ซึ่งจะทำการติดต่อกับคอมพิวเตอร์ผ่านทาง Serial Port



รูปที่ 3.8 วงจรสื่อสารผ่านพอร์ทอนุกรม

3.8 วงจรสร้างฐานเวลาจริง

การสร้างฐานเวลาจริง (Real time clock) จะใช้ไอซีเบอร์ 6242 (U13) ซึ่งจะใช้เป็นชิปที่บอกเวลาจริงให้แก่ระบบ การเชื่อมต่อแสดงดังรูป 3.9 คริสตอลที่ใช้ในการสร้างสัญญาณนาฬิกาใช้ค่าเท่ากับ 32.768 kHz สัญญาณที่ใช้ในการเลือกไอซี 6242 คือ /RTC ซึ่งมีแอดเดรสอยู่ที่ FF80H



รูปที่ 3.9 วงจรสร้างฐานเวลาจริง

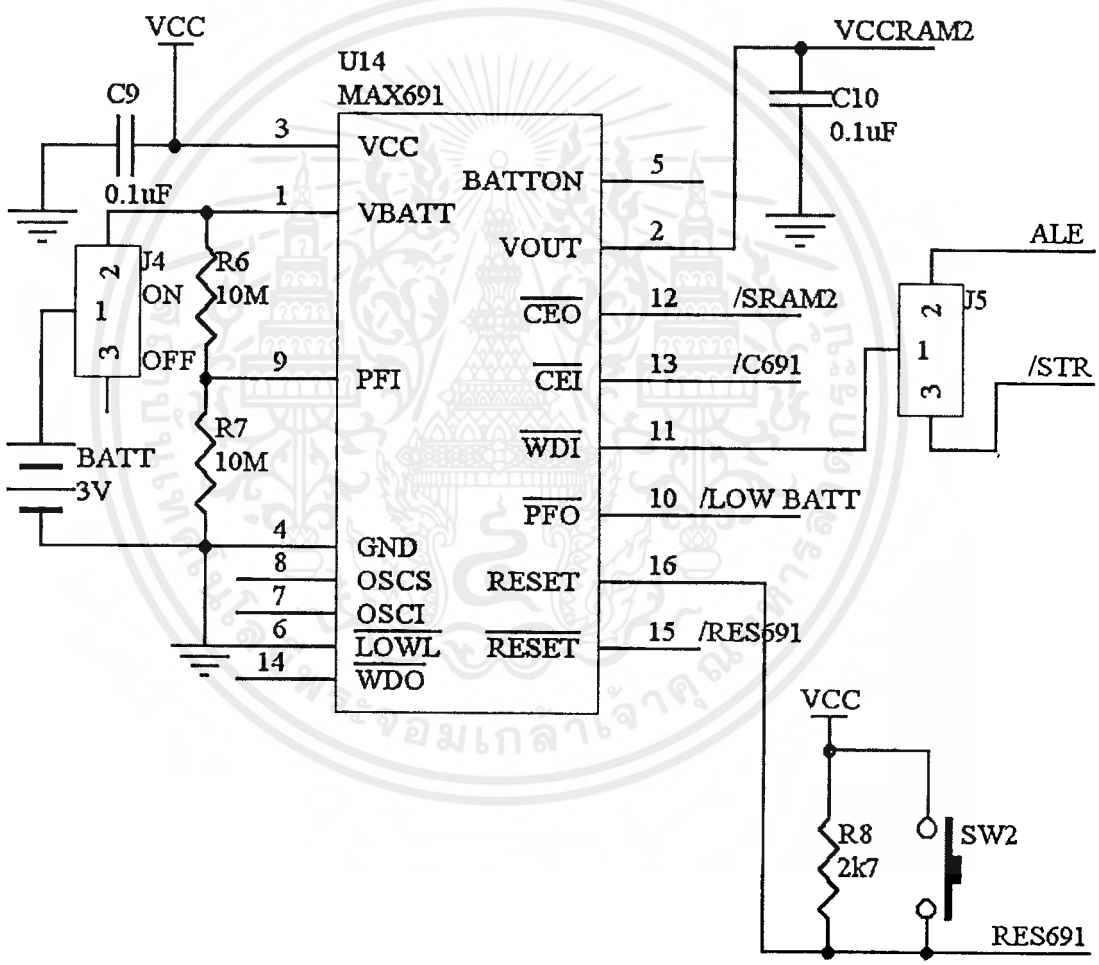
3.9 วงจรวอทช์ดอก (Watch Dog)

วงจรในส่วนนี้จะทำหน้าที่ตรวจสอบการทำงานของระบบ โดยจะใช้ไอซีเบอร์ MAX691 (U14) มีการเชื่อมต่อดังรูปที่ 3.10 วงจรในส่วนนี้จะทำหน้าที่ดูแลระบบเมื่อระบบหยุดทำงาน วงจรในส่วนนี้จะส่งสัญญาณไปรีเซ็ทตัวไมโครคอนโทรลเลอร์ผ่านขารีเซ็ททันทีซึ่งหากไมโครคอนโทรลเลอร์ยังทำงานเป็นปกติ จะต้องทำการส่งสัญญาณมาสโตรบผ่านทาง 74LS138 (/STR ซึ่งมีแอดเดรสอยู่ที่ FFA0H) ภายใน 1.6 วินาที เพื่อให้ตัว MAX691 รู้ว่าระบบยังคงทำงานเป็นปกติ หากไม่ต้องการใช้การทำงานในส่วนนี้สามารถยกเลิกได้โดยใช้จัมเปอร์ J5

นอกจาก MAX691 จะทำหน้าที่ดูแลการทำงานของระบบแล้ว ยังทำหน้าที่จ่ายไฟสำรองให้กับระบบด้วย ซึ่งการทำงานในส่วนนี้ต้องมีแบตเตอรี่สำรอง ซึ่งจะใช้เป็นถ่านลิเธียม 3V ต่อกับขา VBATT ของ MAX691 ซึ่ง MAX691 จะทำการจ่ายไฟสำรองออกทางขา Vout เพื่อนำไปเป็นไฟ

สำรองให้กับแรมและวงจรสร้างฐานเวลาจริง เมื่อแบตเตอรี่ที่ใช้ใกล้จะหมดลง MAX691 จะส่งสัญญาณเตือนผ่านทางขา /PFO ซึ่งต่อไว้กับขา PC2 ของ System 8255 ทำให้สามารถตรวจสอบพลังงานของแบตเตอรี่ผ่านพอร์ทดังกล่าวได้

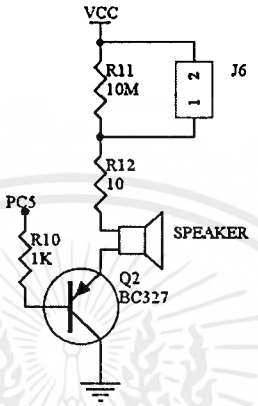
นอกจากนี้ MAX691 ยังสามารถป้องกันการเขียนข้อมูลผิดพลาดเมื่อเกิดสภาวะไฟตกให้กับแรมได้ โดยการนำสัญญาณที่จะนำไปเลือกแรม (ในที่นี้คือ /C691) มาเข้าที่ขา /CEI ก่อน จากนั้นค่อยนำเอาท์พุทที่ขา /CEO (ในที่นี้คือ /SRAM2) ไปต่อกับขา /CS ของแรมอีกที ซึ่งก็จะทำให้เราสามารถป้องกันการเขียนแรมเวลาที่ไฟของระบบหายไป



รูปที่ 3.10 วงจรวอทช์ดอก

3.10 ลำโพง

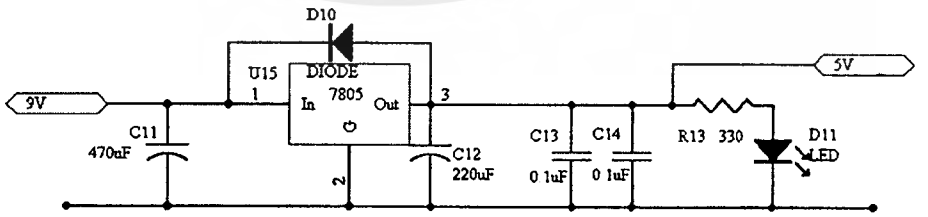
วงจรในส่วนของลำโพงแสดงดังรูป 3.11 ซึ่งจะสั่งให้ลำโพงทำงานได้โดยการสั่งผ่านพอร์ท PC5 ของ System 8255 ซึ่งหากไม่ต้องการให้ลำโพงทำงานก็สามารถทำได้โดยการเซ็ทจัมเปอร์ J6 เพื่อทำการปิดเสียง



รูปที่ 3.11 วงจรขับลำโพง

3.11 วงจรจ่ายไฟเลี้ยง

สำหรับวงจรส่วนนี้จะใช้ไอซีเรกูเลเตอร์เบอร์ 7805 ซึ่งจะจ่ายแรงดันเอาท์พุทคงที่ 5 โวลท์ ไดโอด D9 เป็นตัวป้องกันการจ่ายไฟกลับขั้ว ไดโอด D10 ทำหน้าที่เป็นทางผ่านให้กระแสไหลผ่านตัวไอซี มี LED แสดงสภาวะว่าขณะนี้ระบบได้รับไฟเลี้ยงแล้ว C11-C14 ทำหน้าที่กรองแรงดันให้เรียบอีกครั้ง แสดงดังรูปที่ 3.12

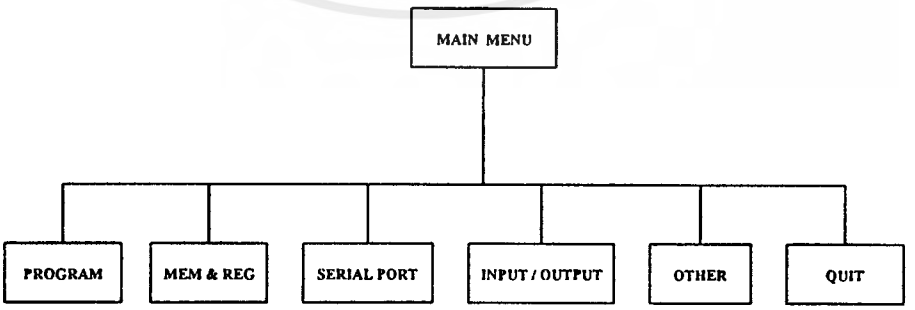


รูปที่ 3.12 วงจรจ่ายไฟเลี้ยง

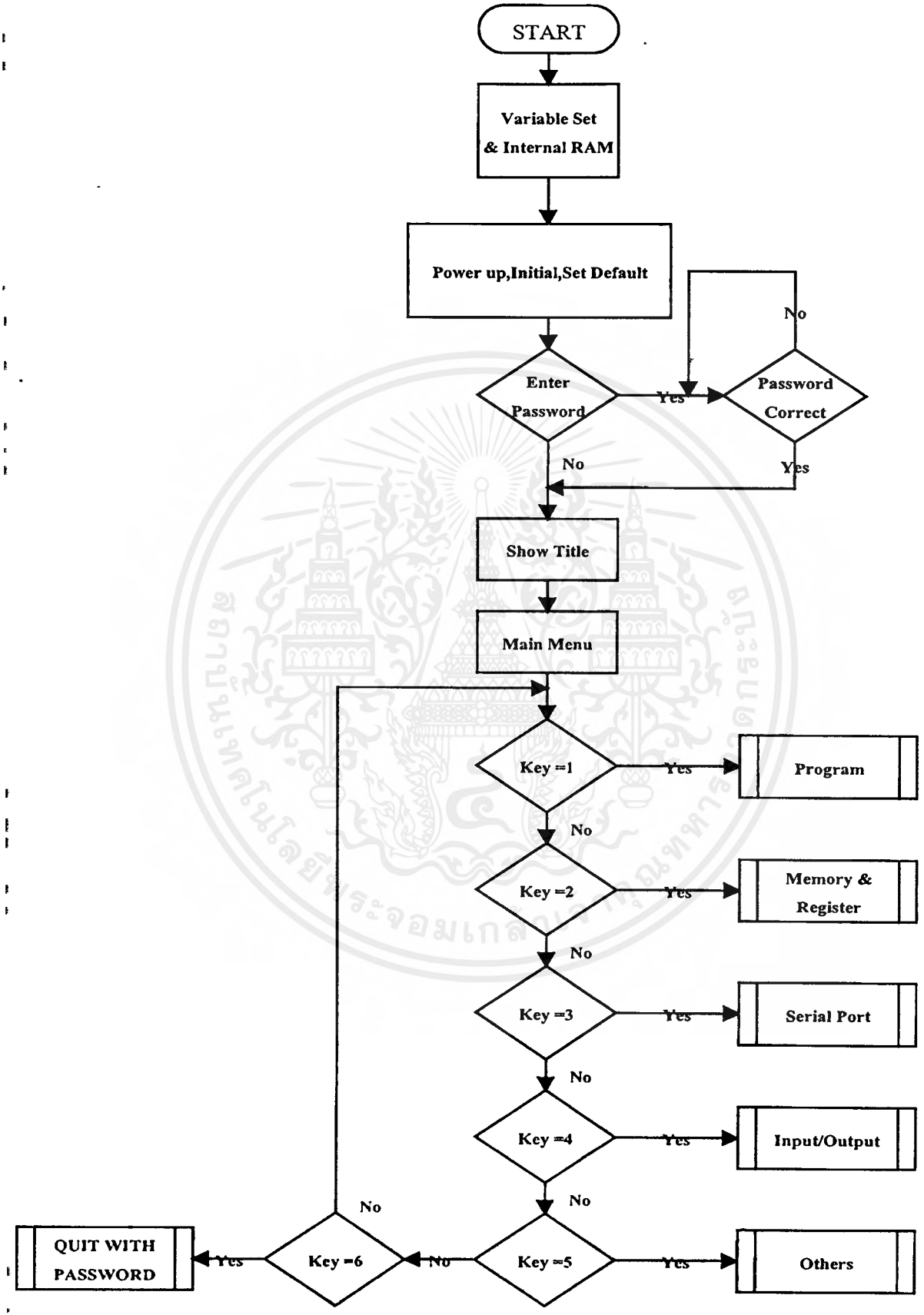
บทที่ 4
โครงสร้างทางด้านซอฟแวร์

เมื่อโปรแกรมเริ่มจะต้องมีการเซ็ทค่าและหน่วยความจำต่าง ๆ จากนั้นจะต้องมีการหน่วงเวลาสำหรับการเริ่มเปิดเครื่อง (Power up Delay) เพราะเวลาเครื่องเปิดมาใหม่ ๆ อุปกรณ์ต่าง ๆ เช่น 8255 ยังไม่พร้อมทำงาน และต้องกำหนดค่าเริ่มต้นต่าง ๆ จากนั้นจะตรวจสอบพาสเวิร์ดของเครื่อง ถ้าผู้ใช้ตอนออกจากเครื่องใส่พาสเวิร์ดเอาไว้ คนอื่นจะไม่สามารถเข้ามาทำงานได้ จากนั้นจะเป็นการแสดงภาพไตเติลของบอร์ด และ เข้าสู่ส่วนเมนูหลักของบอร์ด ซึ่งมีเมนูย่อยให้ผู้ใช้เลือกอยู่ 6 ส่วน คือ

1. PROGRAM ได้แก่ แอสแซมเบลอร์ (Assembler) , อันแอสแซมเบลอร์ (Unassembler) และส่วนรันโปรแกรม (Run Program)
2. MEMORY & REGISTER ได้แก่ หน่วยความจำในส่วนของโปรแกรม, หน่วยความจำในส่วนของข้อมูล , หน่วยความจำภายใน และ รีจิสเตอร์ฟังก์ชั่นพิเศษ
3. SERIAL PORT ได้แก่ การส่งโปรแกรมจากเครื่องคอมพิวเตอร์(Load Hex File) , การส่งโปรแกรมขึ้นเครื่องคอมพิวเตอร์ (Upload Hex File) และ การตั้งความเร็วในการสื่อสารผ่านพอร์ทอนุกรม (Set Baud Rate)
4. INPUT PORT / OUTPUT PORT คือการส่งข้อมูลออกทางพอร์ทของผู้ใช้ (User Port) ผ่านทาง 8255
5. OTHERS ได้แก่ ระบบเวลาจริง (Real Time Clock) และ พอร์ทเครื่องมือ (Tool Port)
6. QUIT ออกจากโปรแกรมด้วยพาสเวิร์ด



รูปที่ 4.1 โครงสร้างของเมนูหลัก



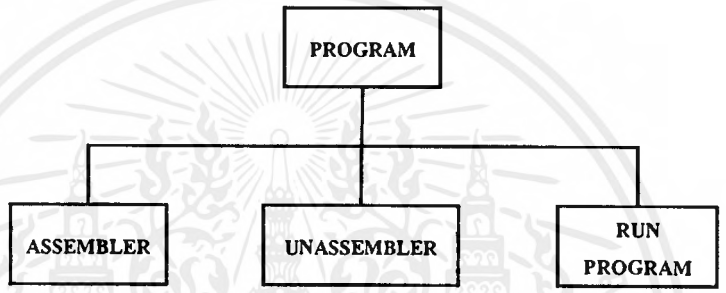
รูปที่ 4.2 โฟลว์ชาร์ทแสดงการทำงานของเมนูหลัก

ซึ่งการทำงานในส่วนของ PROGRAM ประกอบด้วย

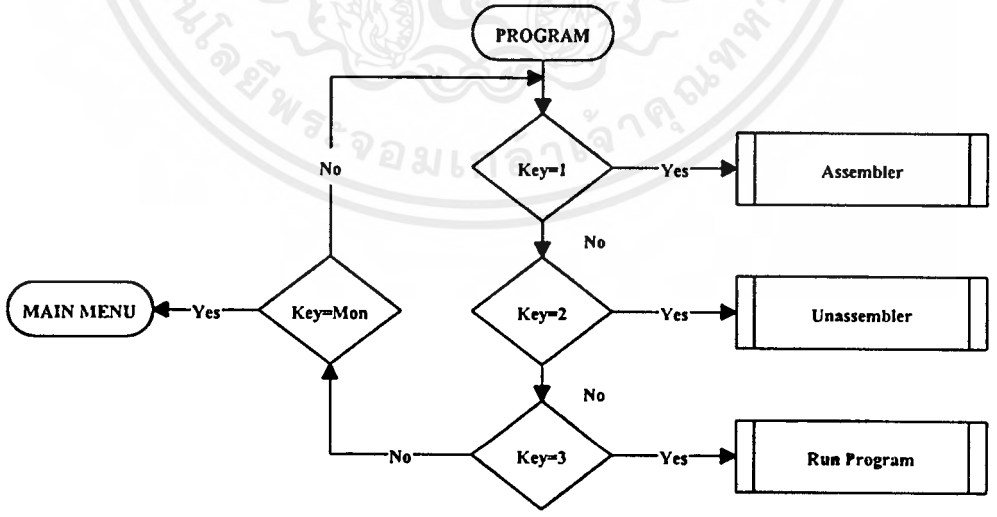
1.โปรแกรมแอสแซมเบลอร์ ใช้ในการเขียนคำสั่งภาษาแอสแซมบลี

2.โปรแกรมอันแอสแซมเบลอร์ ใช้ในการนำคำสั่งภาษาแอสแซมบลีที่ได้เขียนไว้แล้ว มาแก้ไขใหม่

3.รันโปรแกรม ใช้ในการสั่งให้บอร์ดทำการปฏิบัติตามคำสั่ง ที่ได้เขียนไว้แล้ว จากโปรแกรมแอสแซมเบลอร์



รูปที่ 4.3 โครงสร้างของเมนูย่อยในส่วนของโปรแกรม



รูปที่ 4.4 โฟลว์ชาร์ทการทำงานของเมนูย่อยโปรแกรม

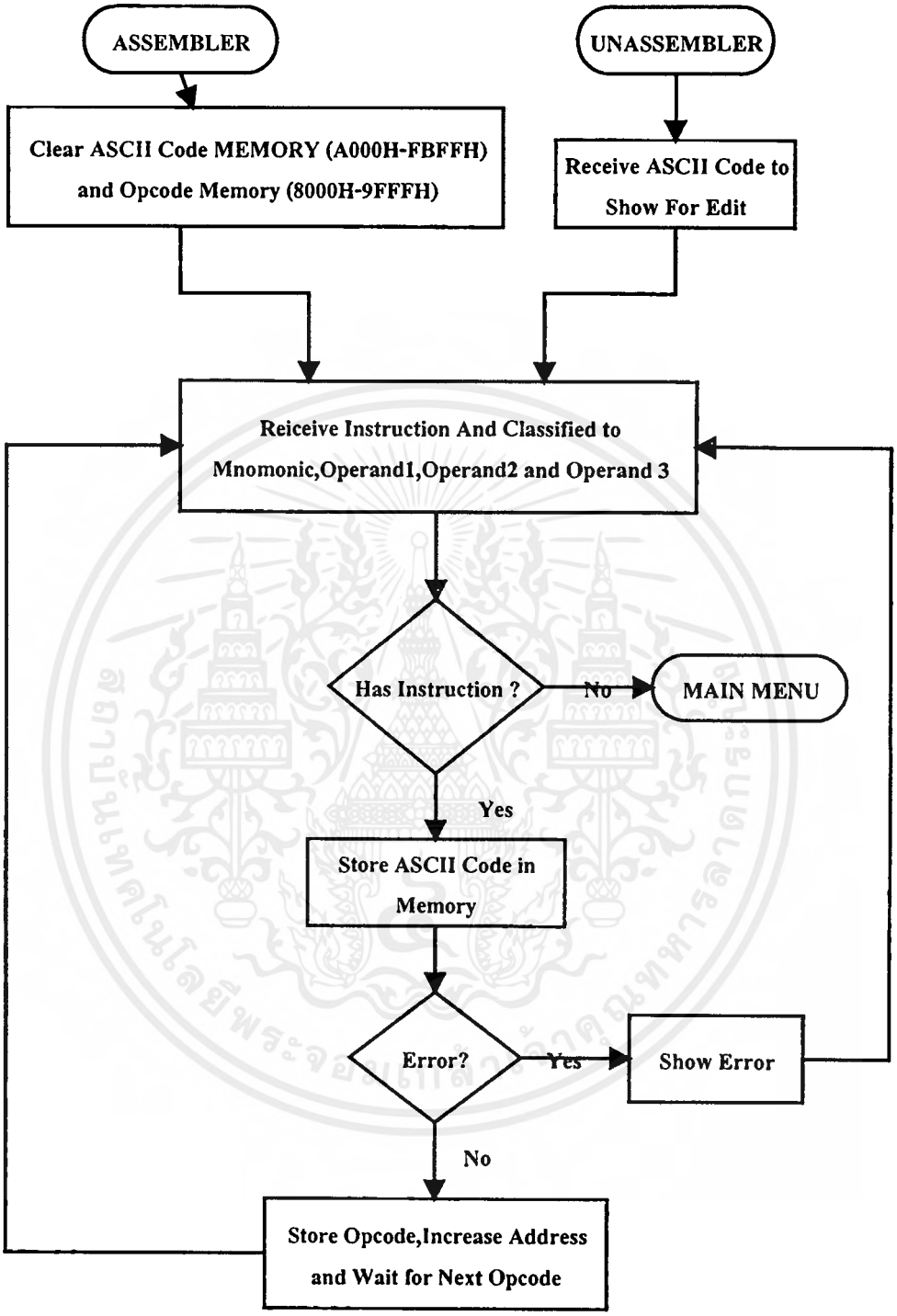
4.1 เมนูย่อยของคำสั่งเกี่ยวกับโปรแกรม

4.1.1 โปรแกรมแอสแซมเบลอร์

เมื่อเข้าสู่โปรแกรมอีดิเตอร์ โปรแกรมจะเคลียร์ส่วนที่ใช้เก็บแอดเดรสหน้าจอ จะแสดงแอดเดรสที่โปรแกรมจะเขียน และมีส่วนให้ผู้ใช้ใส่โปรแกรมที่เป็น นิวมอนิก(Mnomonic) และ โอเปอร์แรน (Operand) และคอมเมนต์ (Comment) ลงไป เมื่อเขียนโปรแกรมจนเสร็จเรียบร้อยแล้ว ต้องการกลับสู่เมนูหลัก สามารถทำได้โดยการกดเอนเทอร์ที่แอดเดรสนั้นโดยไม่ต้องพิมพ์นิวมอนิก ก็จะเป็นการกลับสู่เมนูหลักทันที เมื่อได้พิมพ์โปรแกรมที่เป็นนิวมอนิกและโอเปอร์แรนเรียบร้อยแล้ว กดเอนเทอร์ โปรแกรมแอสแซมเบลอร์จะทำการเก็บรหัสแอสกีที่ตรงกับนิวมอนิกและโอเปอร์แรนนั้นๆไว้ที่แอดเดรส A000H - FBFFH และเช็คความถูกต้องของโปรแกรมทั้งรูปแบบของการใช้คำสั่งและไวยากรณ์ของคำสั่งนั้นๆ ถ้าถูกต้องโปรแกรมแอสแซมเบลอร์ก็จะทำการแปลงเป็นออปโคด (Opcode) หรือภาษาเครื่องแล้วนำไปเก็บที่แอดเดรส 8000H-FBFFH แล้วโปรแกรมแอสแซมเบลอร์จะทำการคำนวณว่าคำสั่งนั้นๆเป็นคำสั่งที่มีความยาวกี่ไบต์ แล้วจะทำการเว้นแอดเดรสที่ได้ใช้ไปโดยอัตโนมัติ และขึ้นแอดเดรสใหม่ ที่ต่อจากจำนวนไบต์ ของคำสั่งนั้นๆทันที แต่ถ้าคำสั่งที่ผู้ใช้พิมพ์ลงไปไม่ถูกต้อง โปรแกรมแอสแซมเบลอร์จะแจ้งให้ผู้ใช้ทราบทันทีว่าโปรแกรมที่เขียนลงไปนั้นไม่ถูกต้อง และจะย้อนกลับมาให้พิมพ์ใหม่โดยทันที

4.1.2 โปรแกรมอันแอสแซมเบลอร์

ส่วนโปรแกรมอันแอสแซมเบลอร์ เป็นการนำโปรแกรมที่ได้พิมพ์ไปแล้วกลับมาแก้ไขใหม่ ซึ่งจะแสดงบรรทัดแรกของโปรแกรม (แอดเดรส 8000H) แล้วจะดำเนินการแก้ไขก็จะเหมือนโปรแกรมอีดิเตอร์ดังขั้นตอนที่แล้ว ต่างกันตรงที่โปรแกรมแอสแซมเบลอร์ จะทำการเคลียร์ส่วนเก็บโปรแกรมเป็นรหัสแอสกี แล้วเริ่มรับโปรแกรมชุดใหม่ แต่อันแอสแซมเบลอร์ จะนำ โปรแกรมที่อยู่ในส่วนเก็บรหัสแอสกี (A000H-FBFFH) ออกมาและรับการแก้ไขโปรแกรม ซึ่งมีโฟล์ชาร์ทการทำงาน แสดงดังรูปที่ 4.5 และรูปแบบของออปโคด ต่าง ๆ แสดงดังรูปที่ 4.6



รูปที่ 4.5 โฟลว์ชาร์ทการทำงานของโปรแกรมแอสแซมเบลอร์ และ อันแอสแซมเบลอร์

Pattern	Operand 1	Operand 2	Operand 3
1	Address 4bits(Hi)+Code 4 bits	Address 8 bit (Low)	-
2	Code 8 bits	Address 8 bits (Hi)	Address 8 bit (Low)
3	Code 8 bits	Address 8 bits	-
4	Code 8 bits	Address 8 bits	Relative Address 8 bits
5	Code 8 bits	Direct Address 8 bits	-
6	Code 8 bits	Direct Address (Source)	-
7	Code 8 bits	Direct Address 8 bits	Immediate Data 8 bits
8	Code 8 bits	Direct Address 8 bits	Relative Address 8 bits
9	Code 8 bits	Immediate Data 8 bits	-
10	Code 8 bits	Immediate Data 8 bits	Relative Address 8 bits
11	Code 8 bits	Immediate Data 8 bits	Immediate Data 8 bits
12	Code 8 bits	Relative Address 8 bits	-
0	Code 8 bits	-	-

รูปที่ 4.6 รูปแบบของออปโคด

4.1.3 การรันโปรแกรม

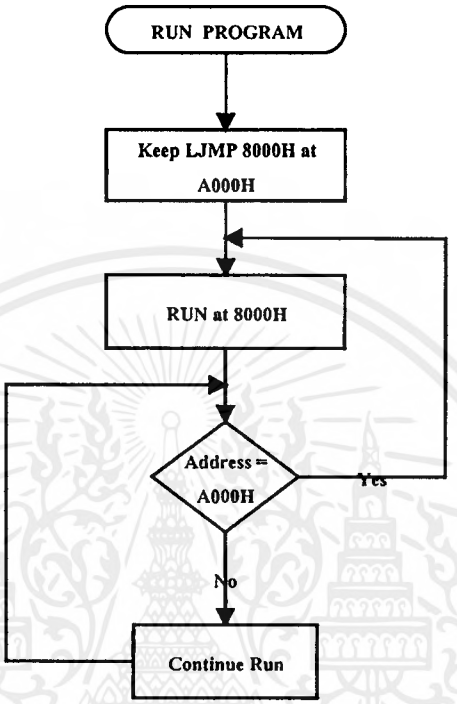
เมื่อเข้าสู่การรันโปรแกรม ซึ่งมีแอดเดรสที่ 8000H-9FFFH โปรแกรมมอนิเตอร์จะทำการเก็บคำสั่ง LJMP 8000H ไว้ที่แอดเดรส A000H เพื่อให้เวลาที่ไมโครคอนโทรลเลอร์ได้ทำการรันโปรแกรมของผู้ใช้เสร็จแล้ว ให้ย้อนกลับไปเริ่มที่แอดเดรส 8000H มิฉะนั้นไมโครคอนโทรลเลอร์จะรันโปรแกรมต่อ และเข้าไปในส่วนของหน่วยความจำ ที่ใช้ในการเก็บรหัสแอสกี (A000H) และเพื่อให้ไมโครคอนโทรลเลอร์ ทำงานในส่วนหน่วยความจำโปรแกรม วนไปเรื่อย ๆ

ดังนั้นเมื่อทำการรันโปรแกรมแล้วถ้าต้องการกลับมาที่เมนูหลัก จะกระทำได้โดยการรีเซตทางฮาร์ดแวร์เท่านั้น การทำงานของการรันโปรแกรมแสดงดัง โฟลว์ชาร์ทรูปที่ 4.7

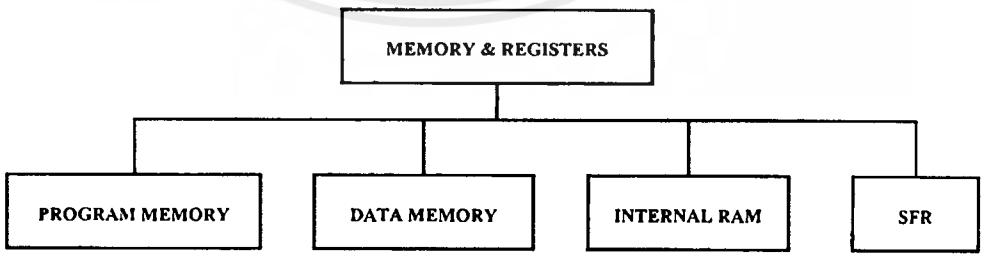
4.2 เมนูย่อยของคำสั่งเกี่ยวกับหน่วยความจำและรีจิสเตอร์

สำหรับการแบ่งหน่วยความจำสำหรับทั้งหมดของบอร์ดซึ่งมีเนื้อที่ทั้งหมด 64 กิโลไบต์ซึ่งจะแบ่ง ออกเป็นส่วนต่างๆ ดังนี้ หน่วยความจำโปรแกรมของผู้ใช้ (User Program Memory) อยู่ที่แอดเดรส 8000-9FFFH และ หน่วยความจำข้อมูลของผู้ใช้ (User Data Memory) อยู่ที่แอดเดรส

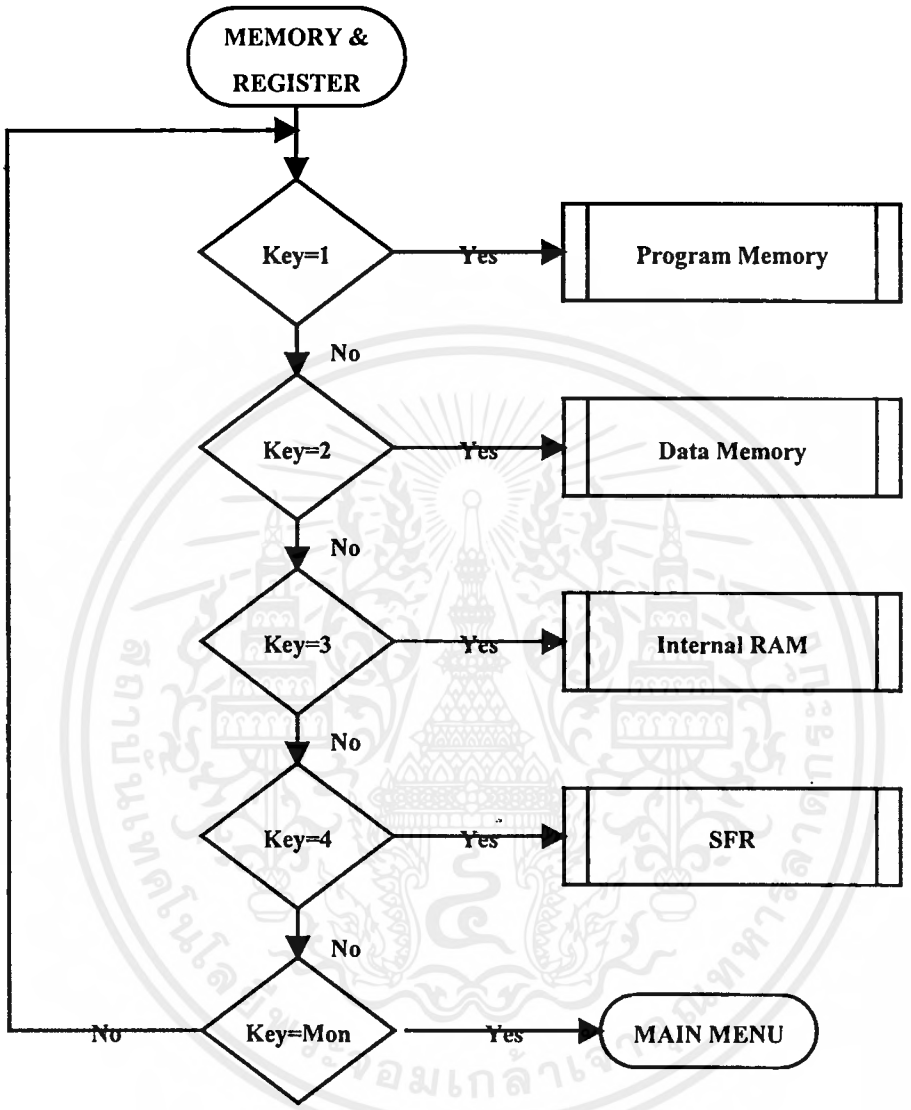
0000-7FFFH และยังมีหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ อีก 128 ไบต์ ซึ่งมีรีจิสเตอร์ฟังก์ชั่นพิเศษรวมอยู่ด้วย จึงสามารถเขียนหน่วยความจำได้ดังรูปที่ 4.8



รูปที่ 4.7 โฟลว์ชาร์ทแสดงการทำงานของการรันโปรแกรม



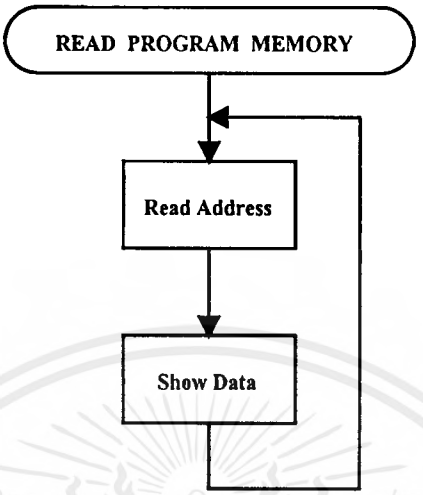
รูปที่ 4.8 การจัดสรรหน่วยความจำของซิงเกิลบอร์ด



รูปที่ 4.9 โฟลว์ชาร์ทการทำงานของเมนูย่อยเกี่ยวกับหน่วยความจำ

4.2.1 หน่วยความจำโปรแกรม

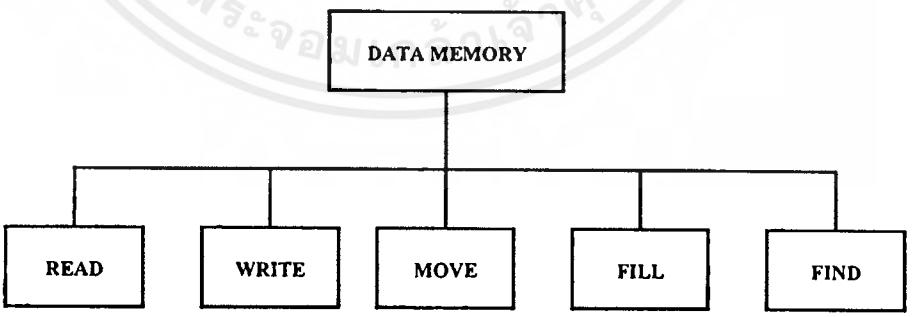
สำหรับหน่วยความจำในส่วนนี้สำหรับผู้ใช้แล้ว สามารถทำได้เพียงแค่อ่านข้อมูลเท่านั้น ไม่อนุญาตให้ผู้ใช้บอร์ดทำการเปลี่ยนแปลงข้อมูลในส่วนนี้โดยตรง แต่จะเปลี่ยนได้โดยการใช้โปรแกรมแอสแซมเบลอร์และ Load Hex File จากพอร์ทอนุกรม เท่านั้น รูปที่ 4.10 แสดงโฟลว์ชาร์ทการอ่านข้อมูลของหน่วยความจำโปรแกรม ผู้ใช้สามารถอ่านข้อมูลบนหน่วยความจำโปรแกรมได้โดยการใส่แอดเดรสที่ต้องการลงไป จากนั้นข้อมูลบนแอดเดรสดังกล่าวก็จะปรากฏ



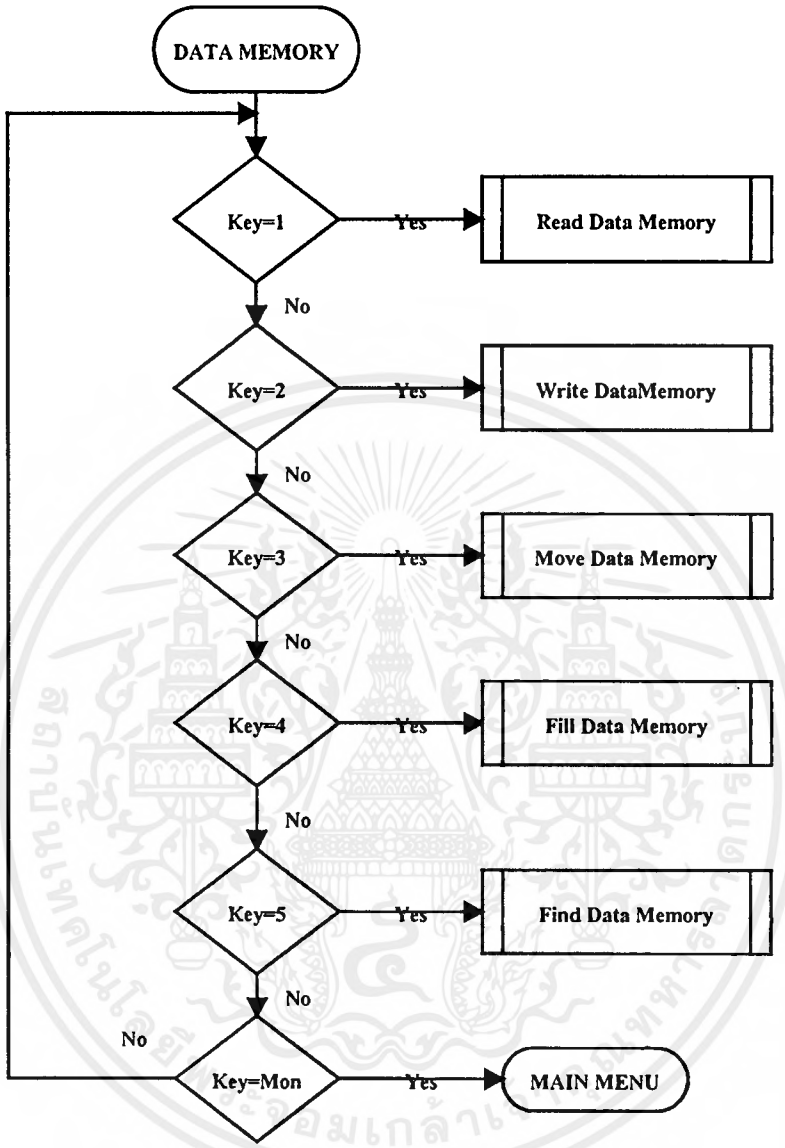
รูปที่ 4.10โฟลว์ชาร์ทแสดงการอ่านข้อมูลของหน่วยความจำโปรแกรม

4.2.2 หน่วยความจำข้อมูล

สำหรับหน่วยความจำในส่วนนี้ สามารถที่จะทำการอ่านหรือเขียนก็ได้ คำสั่งที่เกี่ยวข้องกับหน่วยความจำต่างๆแสดงดังรูปที่ 4.11 และโฟลว์ชาร์ทการทำงานดังรูปที่ 4.12



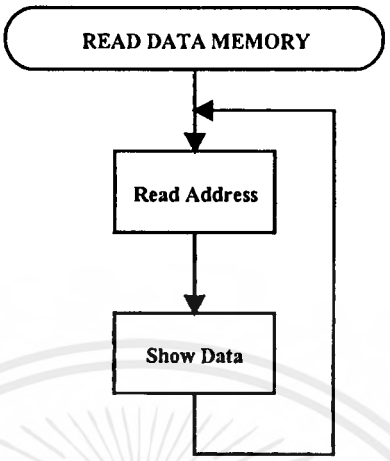
รูปที่ 4.11 เมนูย่อยของคำสั่งเกี่ยวกับหน่วยความจำข้อมูล



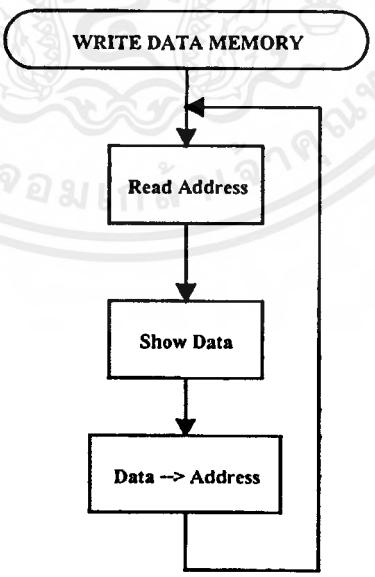
รูปที่ 4.12 โฟลว์ชาร์ทการทำงานของเมนูย่อยเกี่ยวกับหน่วยความจำข้อมูล

การอ่านหน่วยความจำข้อมูลเหมือนกับหน่วยความจำโปรแกรม คือ รับแอดเดรสเข้ามา แล้วแสดงค่าของข้อมูลออกไป ดังโฟลว์ชาร์ทรูปที่ 4.13

การเขียนข้อมูลลงบนหน่วยความจำมีวิธีกระทำ คล้ายกับการอ่านคือจะรับ ค่าแอดเดรสเข้ามาก่อนจากนั้นก็รับค่าข้อมูลที่ต้องเขียนเข้ามา แล้วทำการเขียนลงบนแอดเดรสที่รับเข้ามาก่อนหน้านี้ ซึ่งแสดงโดยโฟลว์ชาร์ทรูปที่ 4.14

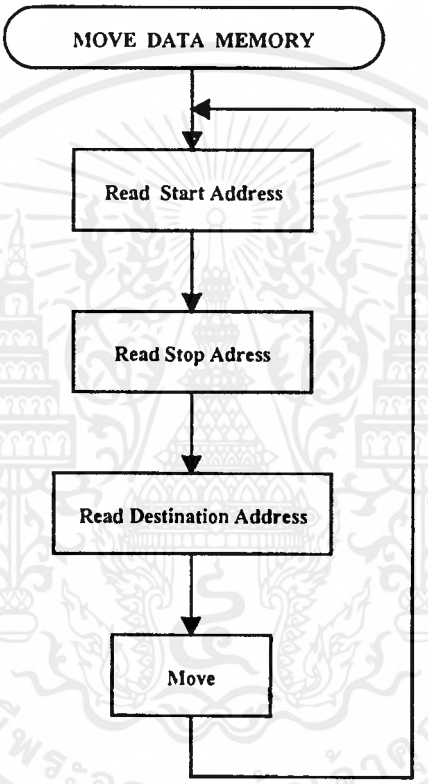


รูปที่ 4.13 โฟลว์ชาร์ทแสดงการอ่านข้อมูลจากหน่วยความจำข้อมูล



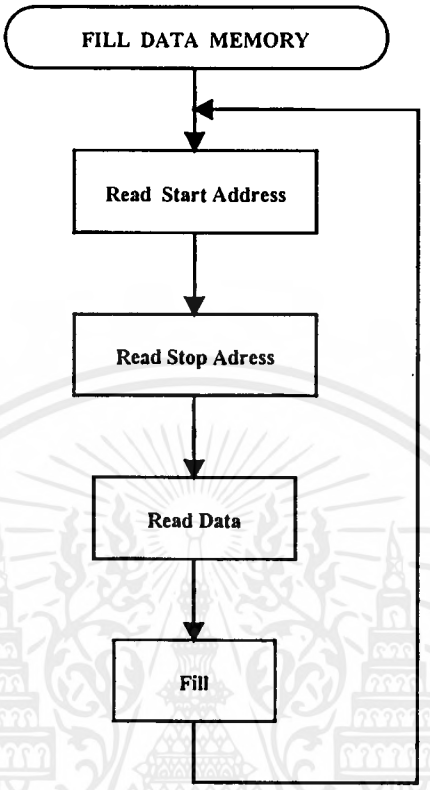
รูปที่ 4.14 โฟลว์ชาร์ทแสดงการเขียนข้อมูลใส่หน่วยความจำข้อมูล

การคัดลอกข้อมูล เป็นการคัดลอกแบบเป็นช่วงหรือ เพียง 1 ไบต์ก็ได้ กระทำโดยการรับแอดเดรส เริ่มต้น และ แอดเดรส สุดท้ายที่จะทำการคัดลอก และ แอดเดรสที่จะทำการคัดลอกไป จากนั้นก็จะคัดลอกข้อมูลทั้งหมดไปไว้ยังที่ที่ต้องการ ดังโฟลว์ชาร์ทรูปที่ 4.15



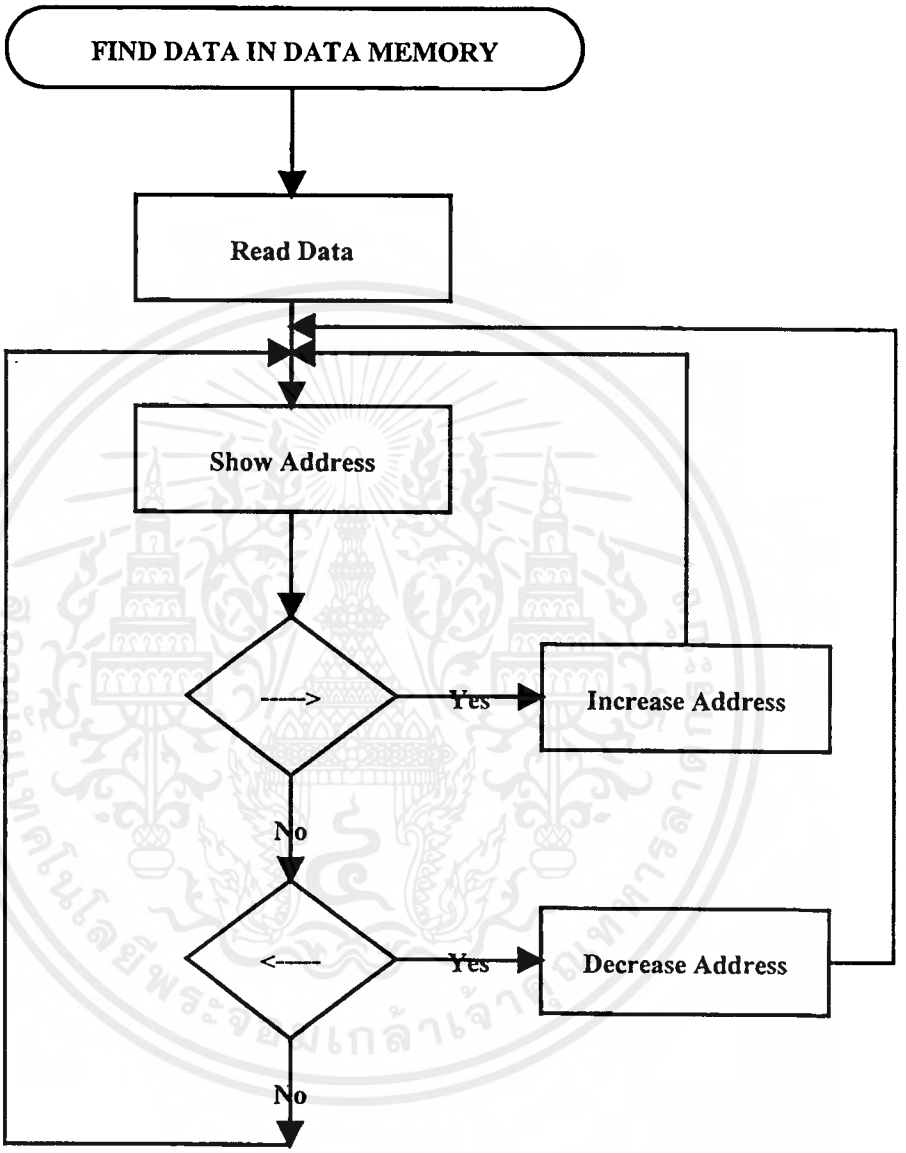
รูปที่ 4.15 โฟลว์ชาร์ทแสดงการคัดลอกข้อมูล

การใส่ข้อมูลในหน่วยความจำข้อมูลทำโดยตอนแรกรับแอดเดรสเริ่มต้น และ แอดเดรสสุดท้าย ที่จะใส่ข้อมูล จากนั้นรับค่าข้อมูลที่จะทำการใส่ลงไปจากนั้น ก็จะใส่ข้อมูลใน แอดเดรส เริ่มต้น ถึงจนถึงแอดเดรส สุดท้าย ดังโฟลว์ชาร์ทรูปที่ 4.16



รูปที่ 4.16 โฟลว์ชาร์ทแสดงการเติมข้อมูลเป็นช่วง

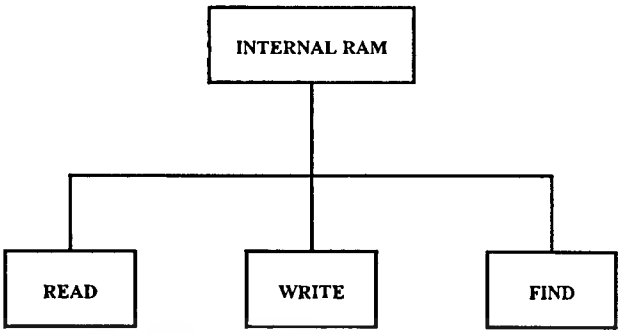
การค้นหาข้อมูลในหน่วยความจำข้อมูล ตอนแรกจะรับข้อมูลมา แล้วจะทำการหาแอดเดรสที่มีข้อมูลตรงกับข้อมูลค่าแรกที่ให้มา ถ้าได้รับคำสั่งให้หาข้อมูลนี้ในตำแหน่งถัดไปจากการกด → ก็จะทำการหาข้อมูลในแอดเดรสถัดไป แต่ถ้าได้รับคำสั่งให้หาข้อมูลนี้ในตำแหน่งก่อนหน้านี้จากการกด ← ก็จะย้อนกลับไปหาที่ตำแหน่งก่อนหน้านี้ ดังโฟลว์ชาร์ท รูปที่ 4.17



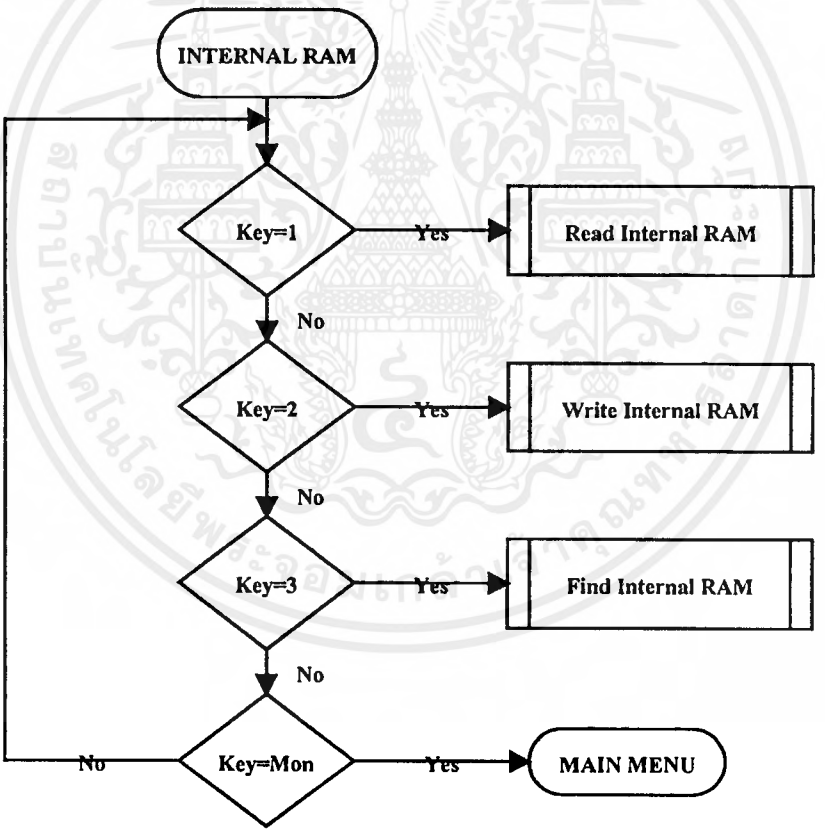
รูปที่ 4.17 โฟลว์ชาร์ทแสดงการค้นหาข้อมูล

4.2.3 หน่วยความจำข้อมูลภายใน

หน่วยความจำข้อมูลภายใน เป็นหน่วยความจำที่มีอยู่ภายในตัวไมโครคอนโทรลเลอร์ มีขนาด 128 ไบต์มีการจัดการดังรูปที่ 4.18 และมีโฟลว์ชาร์ทแสดงการทำงานดังรูปที่ 4.19



รูปที่ 4.18 เมนูย่อยของคำสั่งเกี่ยวกับหน่วยความจำข้อมูลภายใน

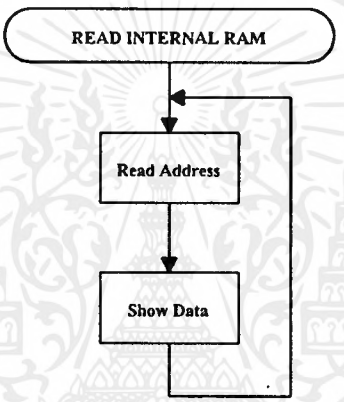


รูปที่ 4.19 โฟลว์ชาร์ทแสดงการทำงานของคำสั่งเกี่ยวกับหน่วยความจำข้อมูลภายใน

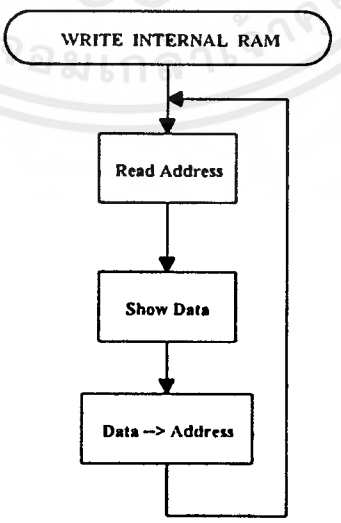
การอ่านหน่วยความจำข้อมูลภายในเหมือนกับหน่วยความจำข้อมูลภายนอก คือ รับแอดเดรสเข้ามา แล้วแสดงค่าของข้อมูลออกไป ดังโฟลว์ชาร์ทรูปที่ 4.20

การเขียนข้อมูลลงบนหน่วยความจำข้อมูลภายในมีวิธีกระทำ คล้ายกับการอ่านคือจะรับ ค่าแอดเดรสเข้ามาก่อนจากนั้นก็รับค่าข้อมูลที่ต้องเขียนเข้ามา แล้วทำการเขียนลงบนแอดเดรสที่เข้ามาก่อนหน้านี้ ซึ่งแสดงโดยโฟลว์ชาร์ทรูปที่ 4.21

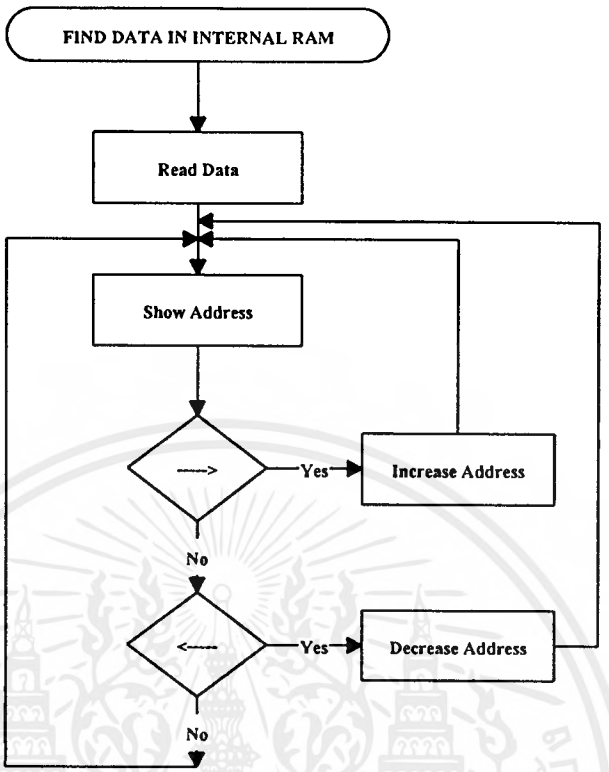
การค้นหาข้อมูลในหน่วยความจำข้อมูล ตอนแรกจะรับข้อมูลมา แล้วจะทำการหาแแอดเดรสที่มีข้อมูลตรงกับข้อมูลค่าแรกที่ให้มา ถ้าได้รับคำสั่งให้หาข้อมูลนี้ในตำแหน่งถัดไปจากการกด → ก็จะทำการหาข้อมูลในแอดเดรสถัดไป แต่ถ้าได้รับคำสั่งให้หาข้อมูลนี้ในตำแหน่งก่อนหน้านี้จากการกด ← ก็จะย้อนกลับไปหาที่ตำแหน่งก่อนหน้านี้ ดังโฟลว์ชาร์ท รูปที่ 4.22



รูปที่ 4.20 โฟลว์ชาร์ทแสดงการอ่านข้อมูลบนหน่วยคามจำข้อมูลภายใน



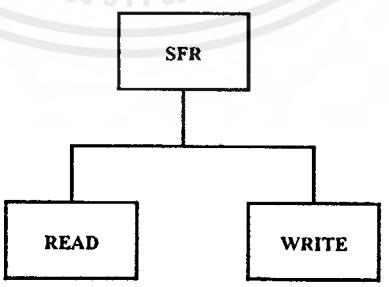
รูปที่ 4.21 โฟลว์ชาร์ทแสดงการเขียนข้อมูลบนหน่วยความจำข้อมูลภายใน



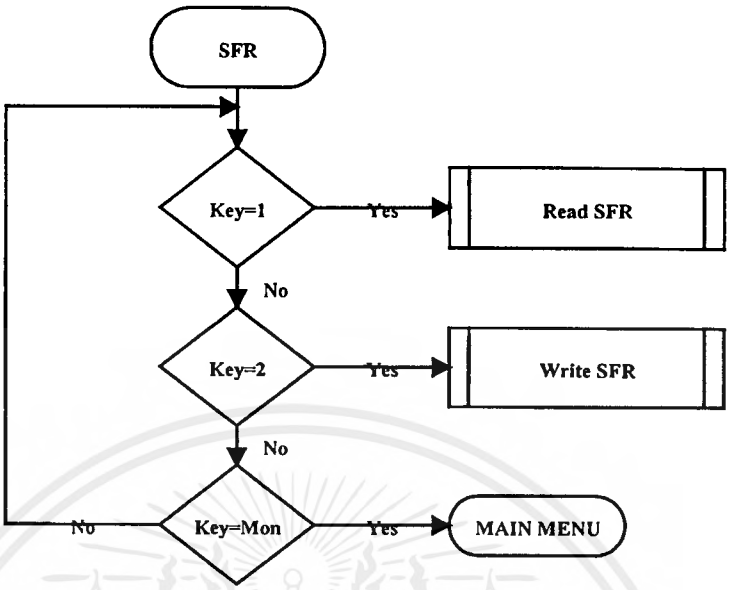
รูปที่ 4.22 โฟลว์ชาร์ทแสดงการหาข้อมูลในหน่วยความจำข้อมูลภายใน

4.2.4 รีจิสเตอร์ฟังก์ชั่นพิเศษ

รีจิสเตอร์ฟังก์ชั่นพิเศษเป็นหน่วยความจำที่มีอยู่ภายในตัวไมโครคอนโทรลเลอร์ มีการจัดการดังรูปที่ 4.23 และมีโฟลว์ชาร์ทแสดงการทำงานดังรูปที่ 4.24



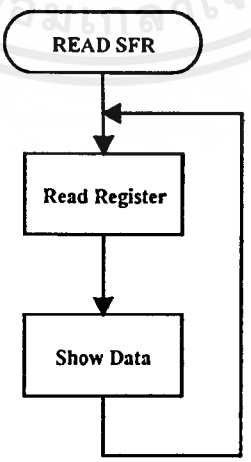
รูปที่ 4.23 เมนูย่อยของคำสั่งเกี่ยวกับรีจิสเตอร์ฟังก์ชั่นพิเศษ



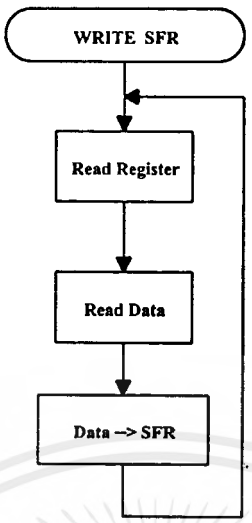
รูปที่ 4.24 โฟลว์ชาร์ทแสดงการทำงานของคำสั่งเกี่ยวกับรีจิสเตอร์ฟังก์ชั่นพิเศษ

การอ่าน รีจิสเตอร์ฟังก์ชั่นพิเศษให้กดคีย์ ←,→ เพื่อเลือกรีจิสเตอร์ แล้วกดเอนเทอร์ ก็จะแสดงค่าของข้อมูลออกไป ดังโฟลว์ชาร์ทรูปที่ 4.25

การเขียนรีจิสเตอร์ฟังก์ชั่นพิเศษให้กดคีย์ ←,→ เพื่อเลือก Register แล้วกดเอนเทอร์ จากนั้นก็รับค่าข้อมูลที่ต้องเขียนเข้ามา แล้วทำการเขียนลงบน SFR ที่รับเข้ามาก่อนหน้านี้ ซึ่งแสดงโดยโฟลว์ชาร์ทรูปที่ 4.26



รูปที่ 4.25 โฟลว์ชาร์ทแสดงการอ่านข้อมูลบนรีจิสเตอร์ฟังก์ชั่นพิเศษ



รูปที่ 4.26 โฟลว์ชาร์ทแสดงการเขียนข้อมูลบนรีจิสเตอร์ฟังก์ชั่นพิเศษ

4.3 เมนูย่อยของคำสั่งเกี่ยวกับการสื่อสารทางพอร์ทอนุกรม

การสื่อสารทางพอร์ทอนุกรมประกอบไปด้วยคำสั่งย่อยๆ อีก 4 คำสั่ง ดังรูปที่ 4.27ประกอบไปด้วย

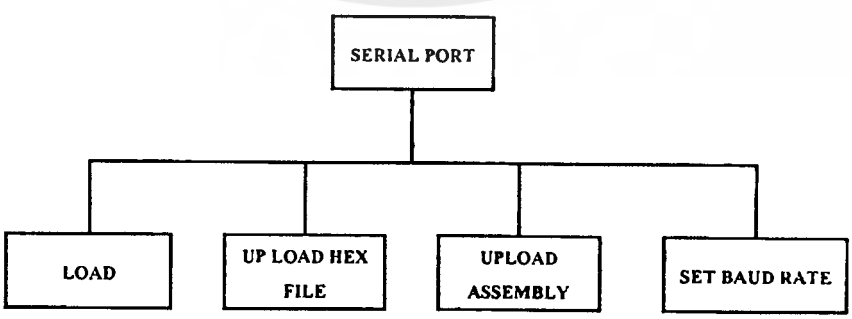
1.การคัดลอกโปรแกรมจากพีซีลงมาสู่บอร์ด (Load Hex File)

2.การคัดลอกโปรแกรมที่เป็น ออปโคด จากบอร์ดสู่พีซี (Upload Hex File)

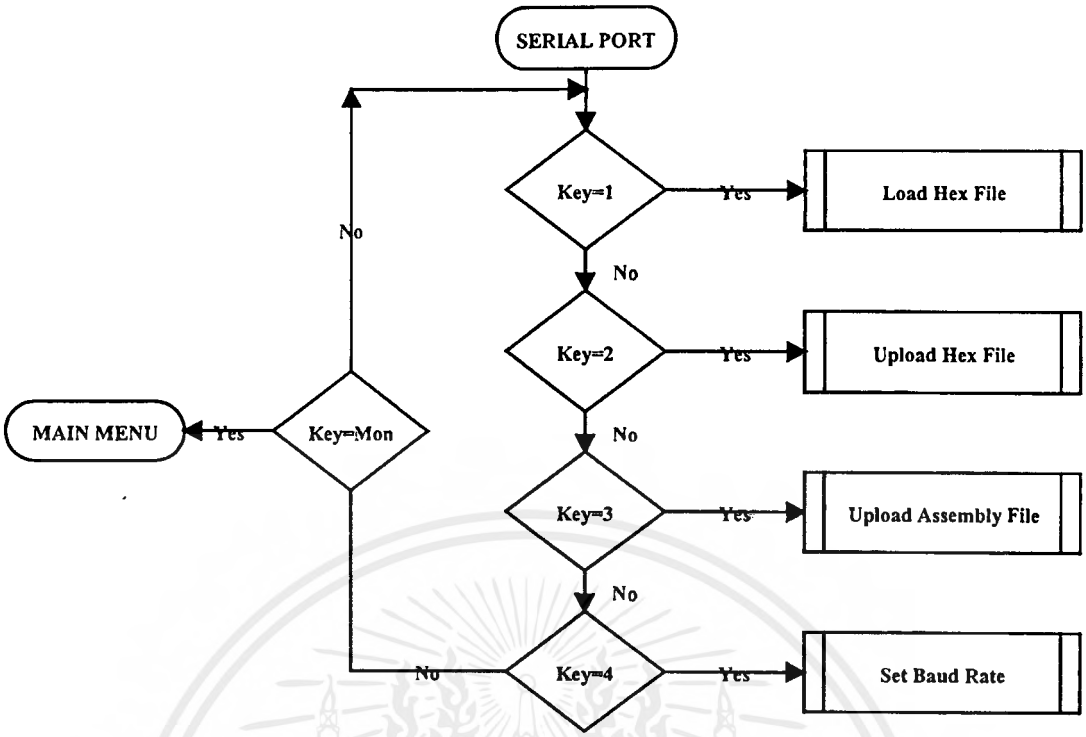
3.การคัดลอกโปรแกรมแอสแซมบลีจากบอร์ดสู่พีซี (Upload ASCII File)

4.การตั้งค่าความเร็วของการติดต่อทางพอร์ทอนุกรม (Set Baud Rate)

ซึ่งเลือกโดยทำตาม Flowchart ดังรูปที่ 4.28



รูปที่ 4.27 เมนูย่อยของคำสั่งเกี่ยวกับการสื่อสารทางพอร์ทอนุกรม



รูปที่ 4.28 โฟลว์ชาร์ทแสดงการทำงานของคำสั่งเกี่ยวกับการสื่อสารทางพอร์ทอนุกรม

4.3.1 การคัดลอกโปรแกรมจากพีซีลงมาสู่บอร์ด

สำหรับการการคัดลอกโปรแกรมจากพีซีลงสู่บอร์ด จะเป็นการรับ Intel Hex File จากพีซีมาลงบอร์ด ในส่วนของหน่วยความจำโปรแกรม (8000H) Intel Hex File คือออปโคด ที่เก็บเป็นรหัสแอสกี ซึ่งมีรูปแบบดังนี้

: BC AAAATTHH.....HHCC

: คือ อักขระเริ่มต้น (Start Character)

BC คือ จำนวนไบต์ ของข้อมูลในบรรทัด (HEX)

ถ้า BC =0 จะเป็นจบข้อมูล (End of File)

AAAA คือแอดเดรสข้อมูลในไบต์แรก

TT คือ ชนิดของข้อมูลในบรรทัดนั้น ๆ

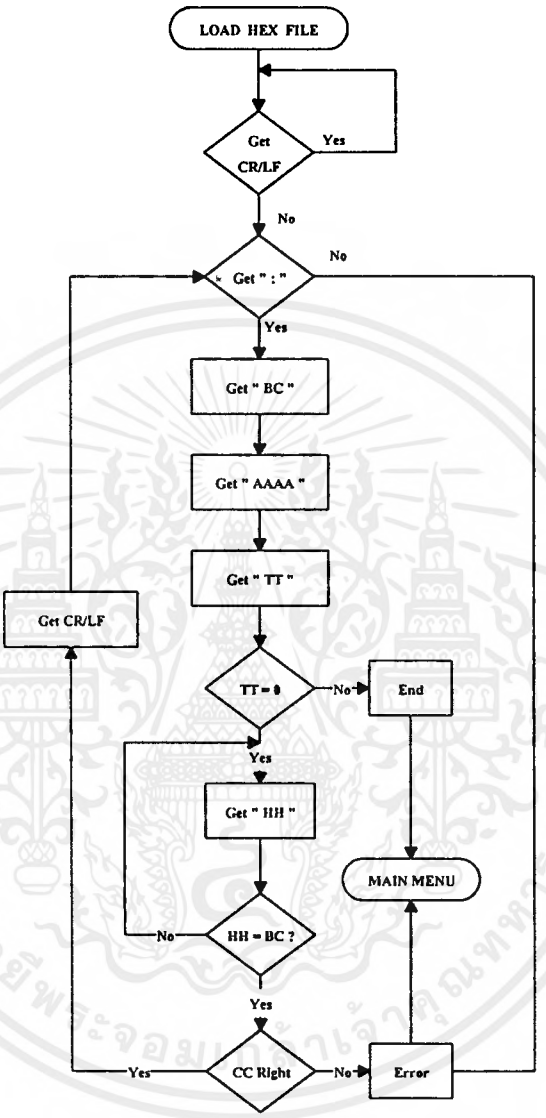
ถ้า TT=0 เป็นข้อมูล

ถ้า TT=1 เป็น การจบข้อมูล

HH คือ ข้อมูลแต่ละไบต์

CC คือ ค่าผลบวกของบรรทัดนั้น ๆโดยจะเป็นค่า ทูคอมพลีเมนต์ ของผลบวกของข้อมูลทุก ๆ ไบต์ ในบรรทัด ซึ่งรวมทั้ง BC , AAA และ TT ด้วย

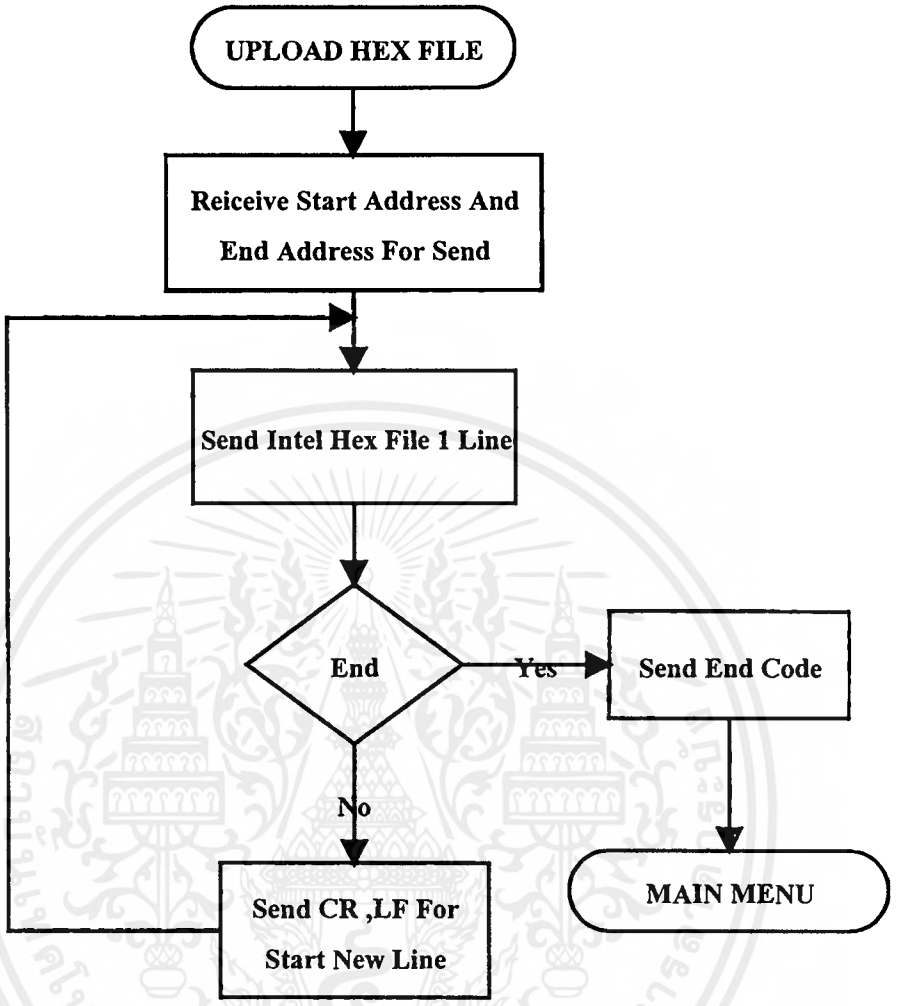
ซึ่งเมื่อคัดลอกเสร็จแล้ว จะทำการเช็ค CC ว่าถูกหรือไม่ ถ้าไม่ถูกแสดงว่าเกิดความผิดพลาดขึ้นและจะแจ้งให้ทราบ การทำงานแสดงโฟลว์ชาร์ทรูปที่ 4.29



รูปที่ 4.29 โฟลว์ชาร์ทแสดงการคัดลอกโปรแกรมจากพีซี

4.3.2 การคัดลอกโปรแกรมจากบอร์ดสู่พีซี

มีวิธีการเหมือนกับการคัดลอกโปรแกรมจากพีซีมาสู่บอร์ด ซึ่งมีรูปแบบของการส่งอยู่ 2 แบบ คือส่งเป็น Intel Hex File และอีกแบบคือส่งเป็นโปรแกรมแอสแซมบลี มีการทำงานดังโฟลว์ชาร์ทรูปที่ 4.30 และ 4.31 ตามลำดับ



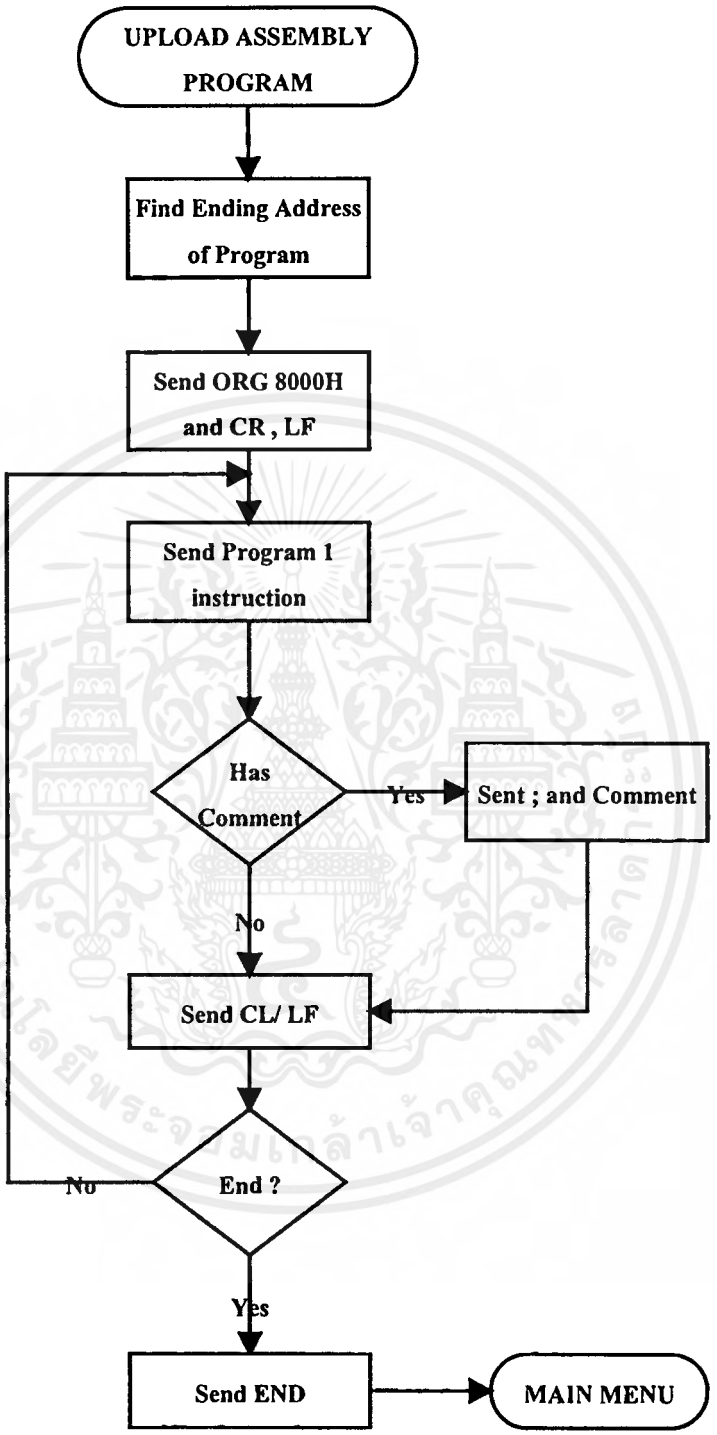
รูปที่ 4.30 โฟลว์ชาร์ทแสดงการคัดลอกข้อมูลจากบอร์ดสู่พีซีแบบ Intel Hex File

4.3.3 การตั้งค่าความเร็วของการติดต่อทางพอร์ทอนุกรม

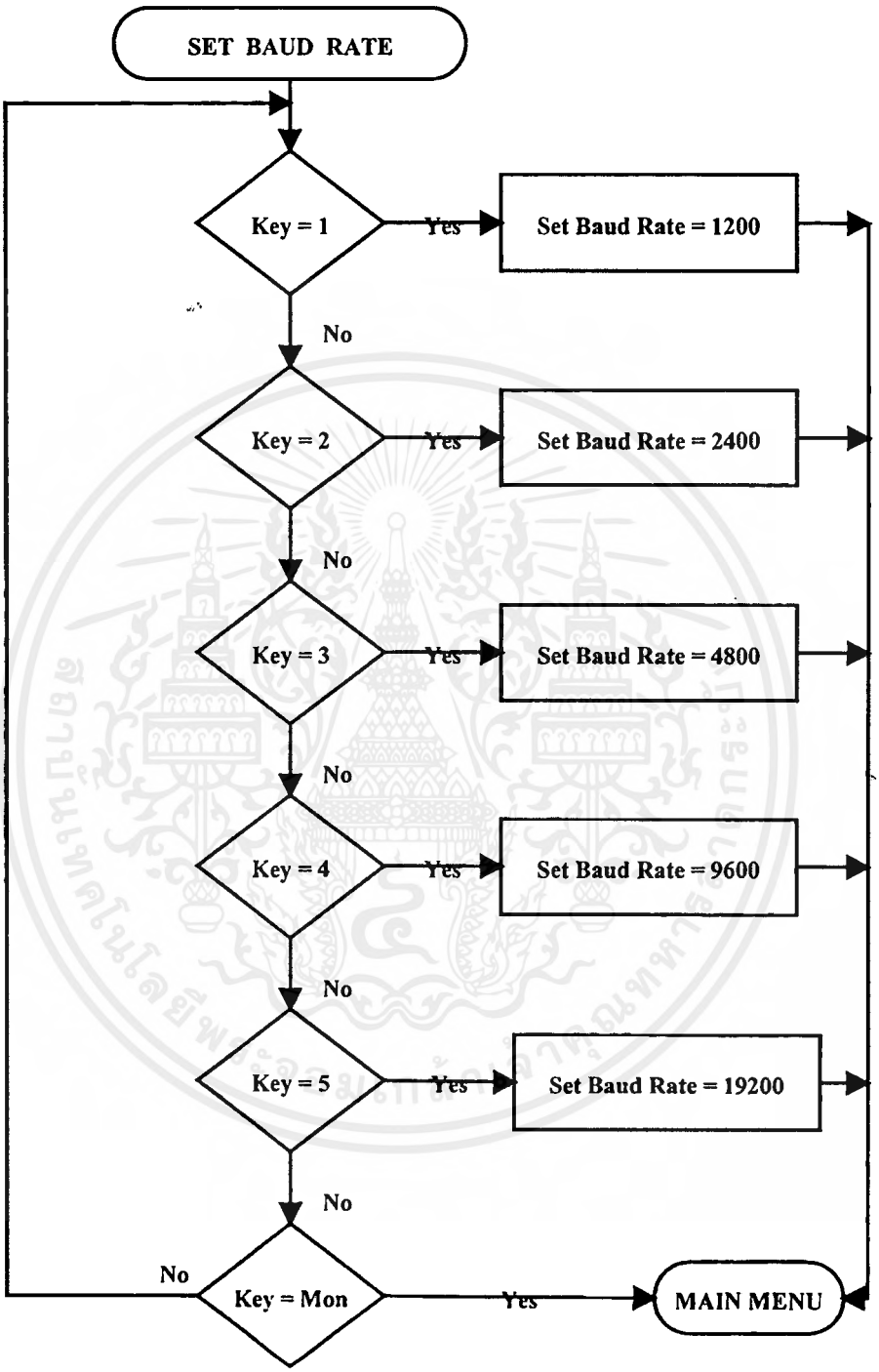
สามารถที่จะทำการตั้งค่าความเร็วของการส่งได้ โดยการรับค่าความเร็วที่ผู้ใช้ต้องการเข้าจากนั้นทำการตั้งค่าความเร็วได้โดยการเซ็ทที่ รีจิสเตอร์TH1 และ บิตSMOD ถ้าไม่ใส่ปกติเป็น 9600

ความเร็ว 1200	TH1 = 0E8H	SMOD = 0
ความเร็ว 2400	TH1 = 0F4H	SMOD = 0
ความเร็ว 4800	TH1 = 0FAH	SMOD = 0
ความเร็ว 9600	TH1 = 0FDH	SMOD = 0
ความเร็ว 19200	TH1 = 0FDH	SMOD = 1

มีโฟลว์ชาร์ทการทำงานดังรูปที่ 4.32



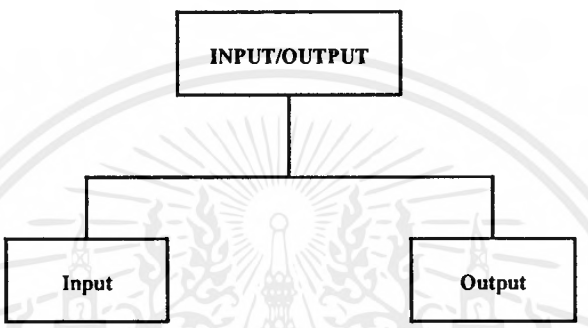
รูปที่ 4.31 โฟลว์ชาร์ทแสดงการคัดลอกข้อมูลจากบอร์ดสู่พีซีเป็นโปรแกรมแอสแซมบลี



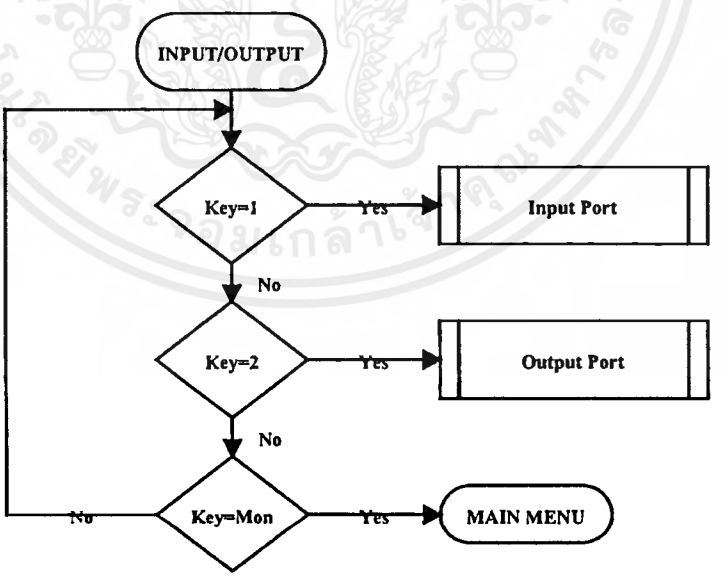
รูปที่ 4.32 การตั้งค่าความเร็วของการติดต่อทางพอร์ทอนุกรม

4.4 เมนูย่อยของคำสั่งอินพุทและเอาท์พุท

เป็นการอินพุทค่าต่าง ๆเข้าทางพอร์ท 8255 ของผู้ใช้เพื่อทำกาประมวลผล หรือ การเอาท์พุทค่าเพื่อนำไปใช้งานโดยผ่านทางพอร์ท 8255 ของผู้ใช้เช่นกัน มีรายละเอียดดังรูปที่ 4.33 และโฟลว์ชาร์ทดังรูปที่ 4.34



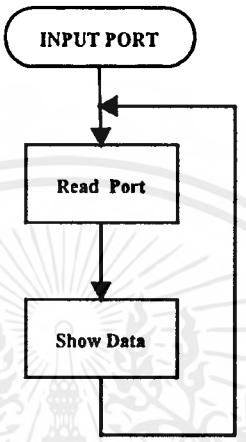
รูปที่ 4.33 เมนูย่อยของคำสั่งอินพุทและเอาท์พุททางพอร์ท 8255



รูปที่ 4.34 แสดงการทำงานของคำสั่งอินพุทและเอาท์พุททางพอร์ท 8255

4.4.1 การอินพุท

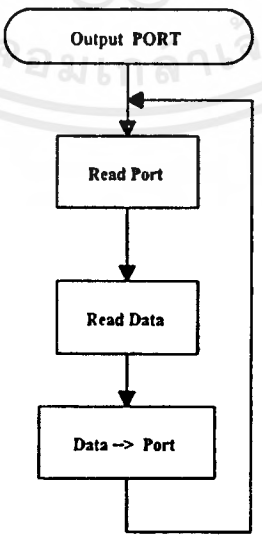
คือการรับค่าเข้ามาทางพอร์ทของ8255 พอร์ท B และ พอร์ท C (เฉพาะ 4 บิตต่ำ) มีการทำงานดังโฟลว์ชาร์ทรูปที่ 4.35



รูปที่ 4.35 โฟลว์ชาร์ทการอินพุทค่าทางพอร์ท 8255

4.4.1 การเอาท์พุท

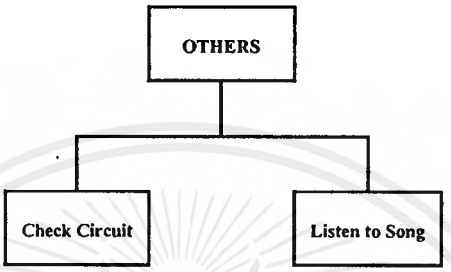
คือการส่งค่าออกไปทางพอร์ทของ8255 พอร์ท A และ พอร์ท C (เฉพาะ 4 บิตบน) มีการทำงานดังโฟลว์ชาร์ทรูปที่ 4.36



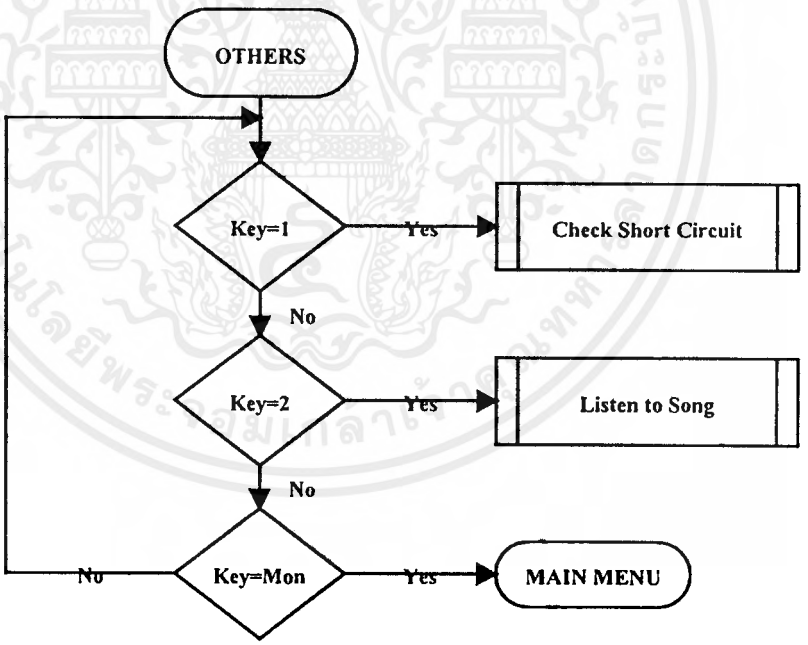
รูปที่ 4.36 โฟลว์ชาร์ทการเอาท์พุทค่าทางพอร์ท 8255

4.5 เมนูย่อยของคำสั่งอื่นๆ

เป็นส่วนของโปรแกรมที่สามารถนำไปใช้ประโยชน์ในด้านอื่นๆ ในที่นี้ได้เขียนไว้ 2 ส่วน คือ อุปกรณ์เช็คการเชื่อมต่อ และ ดนตรี ดัง รูปที่ 4.37 และโฟลว์ชาร์ทการทำงานดังรูปที่ 4.38



รูปที่ 4.37 เมนูย่อยของคำสั่งอื่นๆ

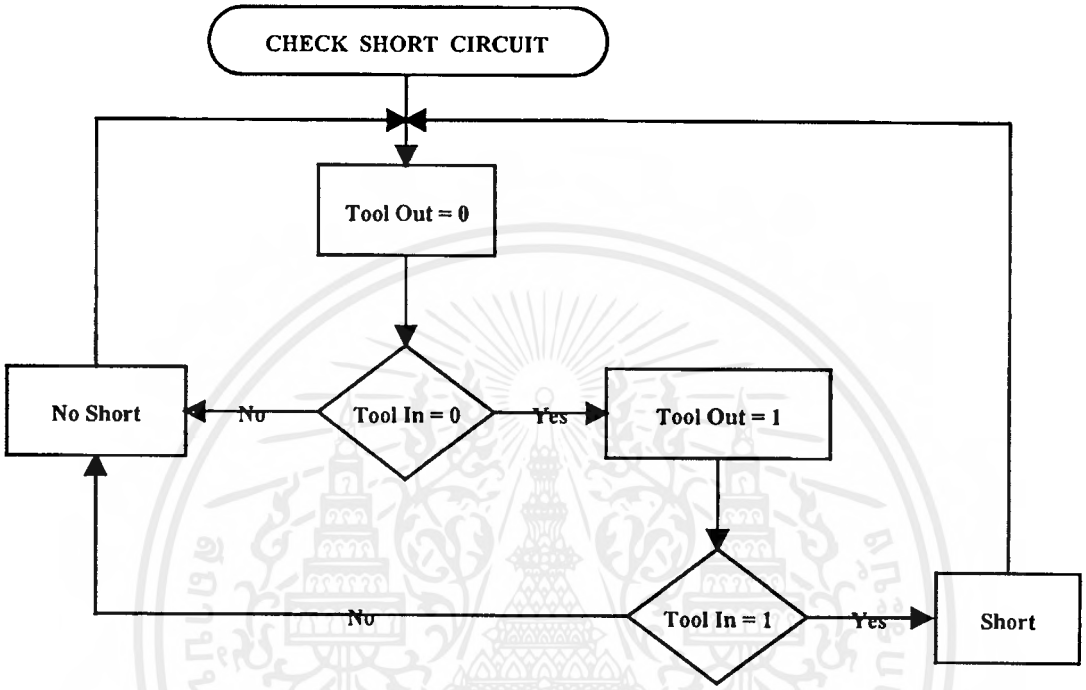


รูปที่ 4.38 โฟลว์ชาร์ทการทำงานของคำสั่งอื่นๆ

4.5.1อุปกรณ์เช็คการเชื่อมต่อ

การตรวจการเชื่อมต่อจะใช้ หลักการง่าย ๆ โดยใช้พอร์ทเครื่องมือ Tool In , Tool Out ในการเช็คคือถ้าวงจรเชื่อมต่อกันเมื่อ Tool Out = 0 แล้ว Tool In จะต้อง = 0 ต่อจากนั้นต้องทำการเช็คให้

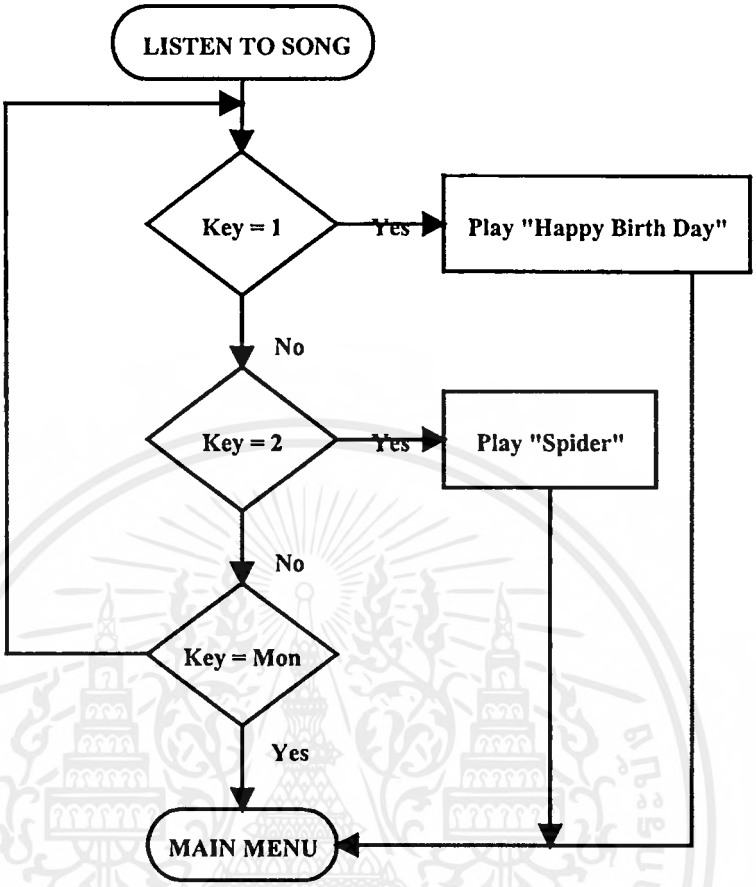
Tool Out = 1 แล้ว Tool In ต้อง เท่ากับ 1 การที่ต้องเช็คทั้งสองสถานะ เพราะหาก ถ้าเช็คสถานะเดียว ถ้าเจอกราวนด์หรือศักดาไฟฟ้าจะทำให้วัดผิดพลาดได้ โฟลว์ชาร์ทการทำงานแสดงดังรูปที่ 4.39



รูปที่ 4.39 โฟลว์ชาร์ทการทำงานของวงจรเช็คการเชื่อมต่อ

4.5.2 ดนตรี

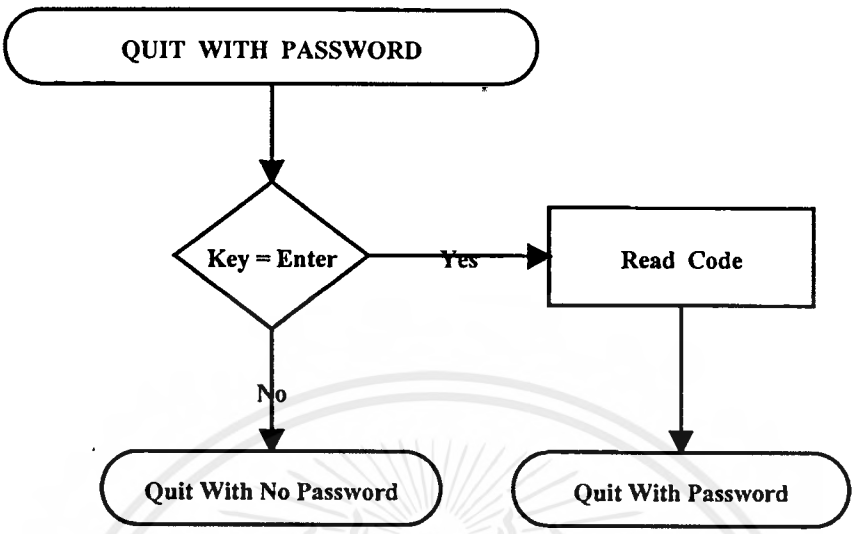
การเล่นเพลง มีขึ้นมาเพื่อความบันเทิงอาจจะนำไปประยุกต์ใช้เป็นนาฬิกาปลุกได้ จะเล่นเพลงโดยใช้รหัสของตัวโน้ตและ รหัสของจังหวะ แล้วจึงจะเล่นเพลงได้



รูปที่ 4.40 โฟลว์ชาร์ทแสดงการเล่นดนตรี

4.6 การออกจากโปรแกรม

การออกจากโปรแกรมการทำงานของโปรแกรมมอนิเตอร์ ทำโดยการกดเลข 6 ที่เมนูหลักจากนั้นถ้ากดเอนเทอร์ออกมาเลยจะเป็นการออกโดยไม่ใช้พาสเวิร์ด แต่ถ้าใส่พาสเวิร์ดก่อนที่จะออกมาแล้ว เวลาเปิดที่เครื่องใช้งานอีกครั้ง จะต้องใส่พาสเวิร์ด ให้ถูกต้องจงจะสามารถเข้ามาใช้งานได้ตามปกติ ถ้าไม่ถูกต้อง จะไม่สามารถใช้เครื่องได้เลย ซึ่งเป็นการป้องกันไม่ให้บุคคลอื่น มาใช้งานบอร์ดในกรณีที่ไม่อนุญาตให้ผู้อื่นใช้งาน หรือ ไม่ต้องการให้บุคคลอื่นมาลบหรือเขียนทับโปรแกรมของตน โฟลว์ชาร์ทการทำงานแสดงดังรูปที่ 4.41



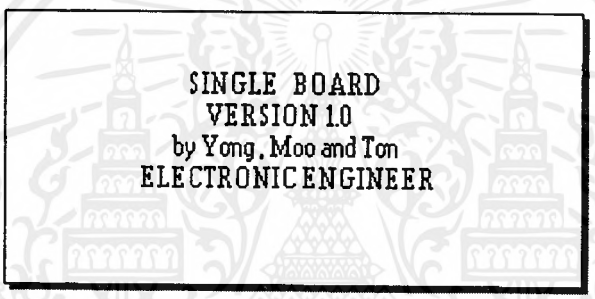
รูปที่ 4.41 โฟลว์ชาร์ทแสดงการป้องกันการใช้งานบอร์ด

บทที่ 5

ผลการทดลอง

5.1 จอภาพเมื่อเริ่มการใช้งาน

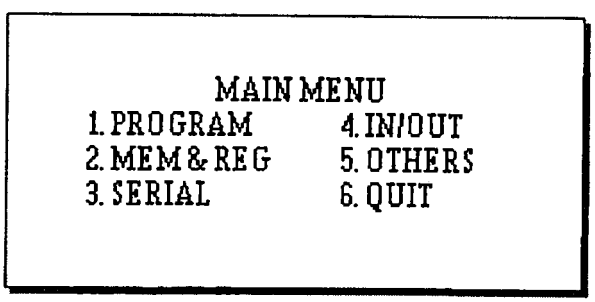
เมื่อเริ่มทำการจ่ายไฟให้กับซิงเกิลบอร์ด จะเกิดภาพของไตเติลขึ้นบนจอแอลซีดี ดังรูปที่ 5.1 ประมาณ 2 วินาที



รูป 5.1 จอภาพเริ่มต้นใช้งาน

จากนั้นจอภาพจะตัดเข้า สู่ภาพเมนูหลัก ดังรูป 5.2 ซึ่งตัวบอร์ดจะรอรับคำสั่งจากผู้ใช้ ซึ่งจะมีตัวเลือกให้ผู้ใช้ 6 ตัวเลือกคือ

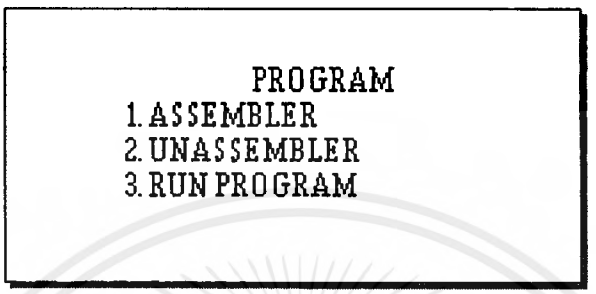
1. Program	2. Memory and Register
3. Serial	4. Input/Output
5.Others	6.Quit



รูป 5.2 เมนเมนู

5.2 การใช้ โปรแกรมแอสแซมเบลอร์

จากเมนูหลัก เมื่อทำการเลือกข้อ 1 จะเป็นการใช้ในส่วนของโปรแกรมแอสเซมเบลอร์ โดยจะเกิดเมนูย่อยดังรูปที่ 5.3



รูป 5.3 เมนูย่อยของโปรแกรม

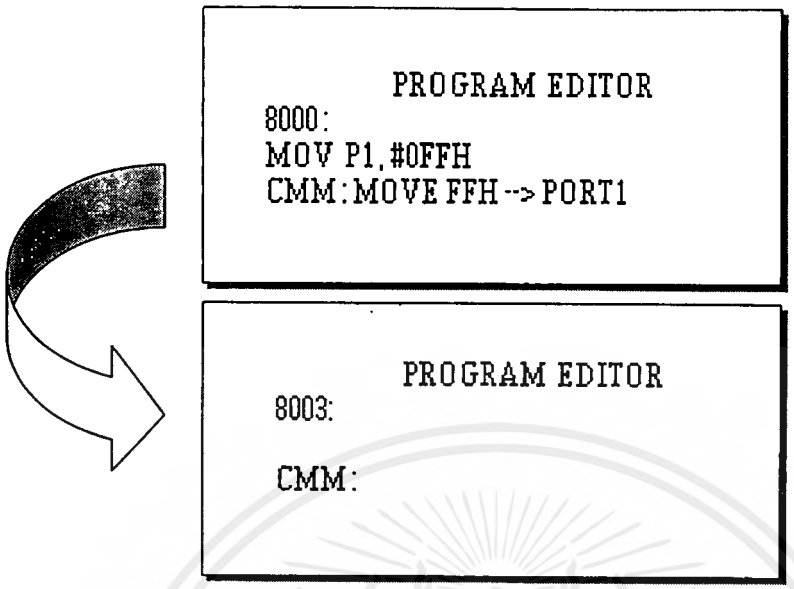
5.2.1 ASSEMBLER

เมื่อเราทำการเลือกข้อ 1 จะเป็นการเรียกใช้อีดิตเตอร์ โดยผู้ใช้สามารถพิมพ์โปรแกรมเป็นภาษาแอสเซมบลีได้ทันที โดยเริ่มต้นที่ แอดเดรส 8000H ดังรูปที่ 5.4

รูป 5.4 โปรแกรมอีดิเตอร์

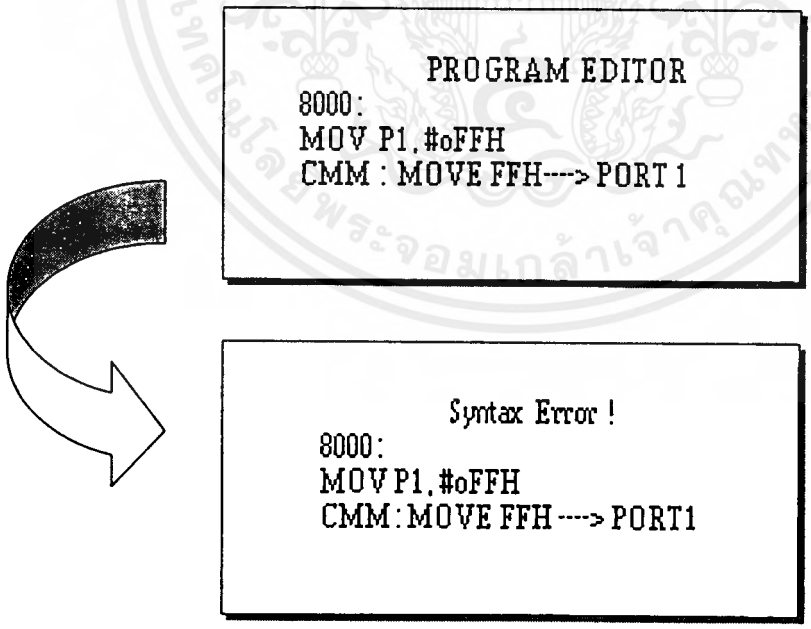
CMM : เป็นที่ว่างสำหรับให้ผู้ใช้เขียนหมายเหตุของโปรแกรมบรรทัดนั้น ๆ

เมื่อทดลองทำการเขียนโปรแกรมสั้น ๆ โดยเขียน MOV P1,#0FFH จากนั้นให้กด เอนเทอร์ เนื่องจากคำสั่ง MOV P1,#0FFH เป็นคำสั่งที่มีความยาว 3 ไบต์ดังนั้น แอดเดรสถัดไปที่แสดงบนจอภาพก็จะเป็น 8003H



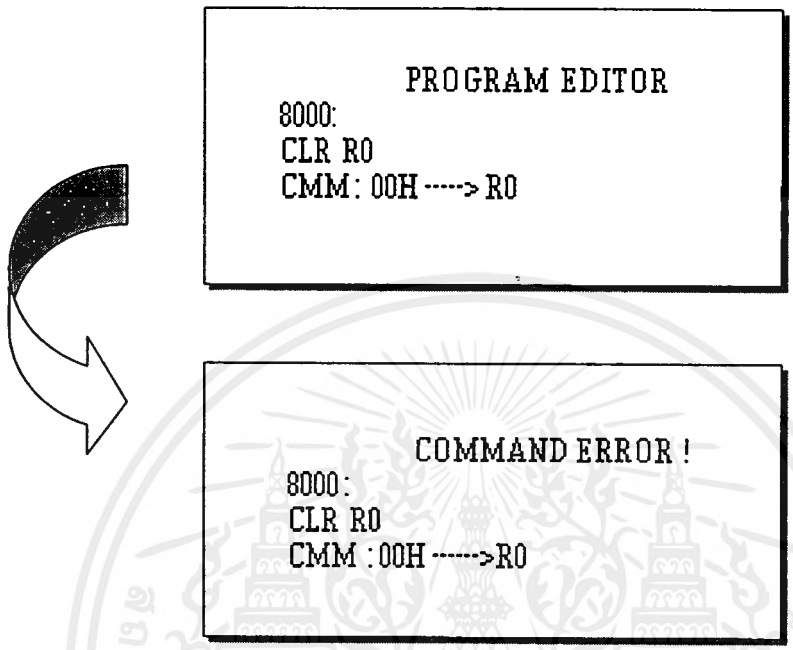
รูป 5.5 การเขียนแอสเซมบลี

ทดลองโปรแกรมให้ผิดพลาดโดยเขียน 0FFH เป็น oFFH เมื่อทำการ กดเอนเทอร์จะปรากฏข้อความ Syntax Error! แสดงว่า ผู้ใช้เขียนคำสั่งผิดไวยากรณ์



รูป 5.6 การเขียนโปรแกรมผิดไวยากรณ์

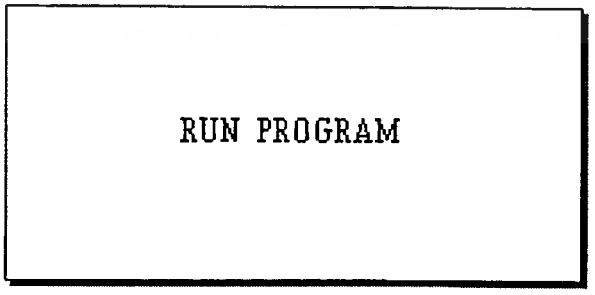
ต่อไปทำการทดลองป้อนโปรแกรมด้วยคำสั่งที่ไม่มีในชุดคำสั่งของ MCS51 โดยเขียนคำสั่ง CLR R0 ซึ่งเมื่อทำการกด เอนเทอร์ จะปรากฏข้อความ Command Error! ดังรูป 5.7



รูป 5.7 การเขียนโปรแกรมโดยใช้คำสั่งผิด

5.2.2 RUN PROGRAM

ทำการป้อนคำสั่งลงไป 2 คำสั่งคือ MOV P1, 0AAH จากนั้นกด เอนเทอร์ 2 ครั้ง จอภาพจะย้อนกลับมาที่ เมนเมนู ทำการเลือกข้อ1 อีกครั้ง เพื่อทำการเรียกใช้โปรแกรมที่ได้เขียนเอาไว้ กดข้อ3 คือ RUN PROGRAM เพื่อทำการทดสอบโปรแกรม ตัวบอร์ดจะทำการรัน โปรแกรมที่ได้พิมพ์ไว้ และแสดงข้อความ RUN PROGRAM ดังรูป 5.8



รูป 5.8 แสดงภาพขณะทำการรันโปรแกรม

ทำการวัดศักดาที่ขา 1 ของพอร์ท 1 ปรากฏว่า ที่ขาของพอร์ท 1 ได้ศักดา 5 V สลับกับ 0 V ซึ่งตรงกับค่าที่ได้ทำการเขียนโปรแกรมไว้

จากนั้นทำการเลิกรันโปรแกรมโดยการกด รีเซท

5.2.3 UNASSEMBLER

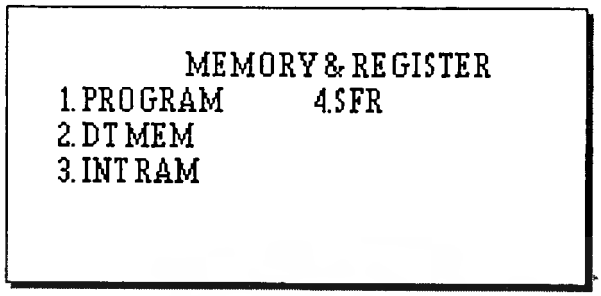
หลังจากทำการรีเซทออกจากกรันโปรแกรม ทำการทดลองดึงออปโคดของคำสั่งกลับมาโดยโปรแกรม UNASSEMBLER โดยการเลือก ข้อ2 ซึ่งบอร์ดจะแสดงหน้าจอขณะ ทำการ UNASSEMBLER ดังรูปที่ 5.9

รูป 5.9 แสดงภาพขณะทำการ UNASSEMBLER

จากนั้นจอภาพก็จะกลับไป ที่หน้าจอของโปรแกรมอิดิเตอร์ ดังรูป 5.5 ซึ่งก็จะได้ชุดคำสั่งเป็น นิวมอนิก และโอเปอร์แรนกลับมาดังเดิม

5.3 การติดต่อกับหน่วยความจำและรีจิสเตอร์

จากเมนูหลักเมื่อทำการเลือกข้อ 2 ก็จะเป็นคำสั่งเกี่ยวกับการจัดการข้อมูลบนแรมและ รีจิสเตอร์ ดังรูปที่ 5.10 ปรากฏคำสั่งย่อย



รูป 5.10 เมนูย่อยของคำสั่ง MEM&REG

อีก 4 คำสั่งคือ 1. Program Memory

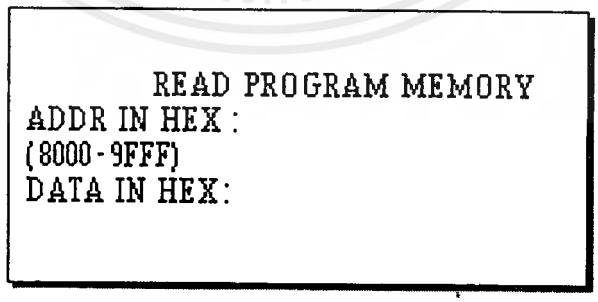
2. Data Memory

3. Internal Ram

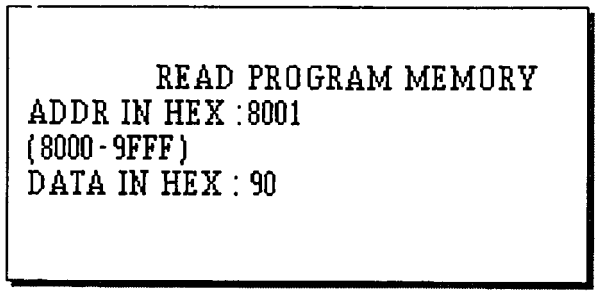
4. SFR (Special Function Register)

5.3.1 Program Memory

เมื่อทำการเลือกข้อ1 ของเมนูย่อย จะเป็นคำสั่งสำหรับการอ่านค่าที่อยู่ใน หน่วยความจำข้อมูล(เป็นหน่วยความจำโปรแกรม ผู้ใช้บอร์ด แต่เป็น หน่วยความจำข้อมูล ของตัวบอร์ด) จากขั้นตอนที่แล้วที่ทำการเขียนคำสั่ง MOV P1,#0AAH ซึ่งมีออปโคดเป็น 75H , 90H และ AAH ทดลองทำการอ่านค่าของโปรแกรมที่อยู่ที่แอดเดรส 8001H ซึ่งได้ค่า 90H ตรงกับออปโคด ที่ได้จากโปรแกรมแอสแซมเบลอร์



รูป 5.11 คำสั่งในการอ่านข้อมูลหน่วยความจำโปรแกรม

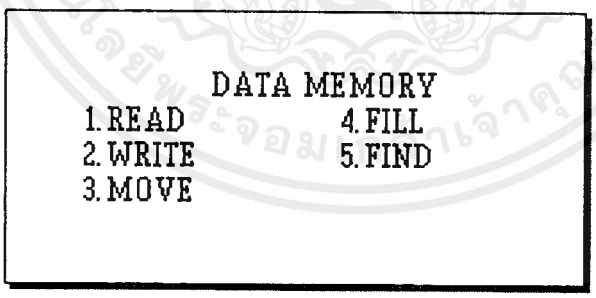


รูป 5.12 ผลของการอ่านโปรแกรมที่แอดเดรส 8001H

5.3.2 Data Memory

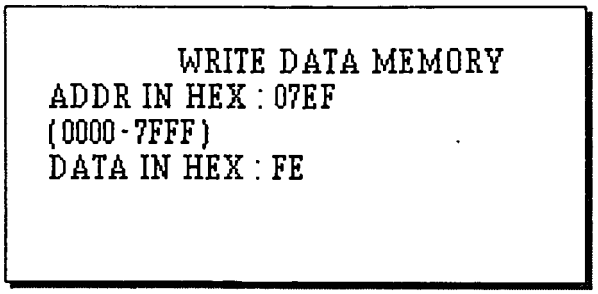
เมื่อทำการเลือกข้อ 2 ซึ่งเป็นคำสั่งที่เกี่ยวกับ หน่วยความจำข้อมูล พบคำสั่งย่อย อีก 6 คำสั่ง ดังรูป 5.13 คือ

1. READ
2. WRITE
3. MOVE
4. FILL
5. FIND



รูป 5.13 เมนูย่อยของคำสั่งเกี่ยวกับ หน่วยความจำข้อมูล

ตอนแรกทดลองเขียนข้อมูลลงบนแอดเดรสที่ต้องการ โดยเลือกข้อ 2 ทำการเขียนข้อมูลFEH ลงบน แอดเดรส 07EFH ดังรูปที่ 5.14

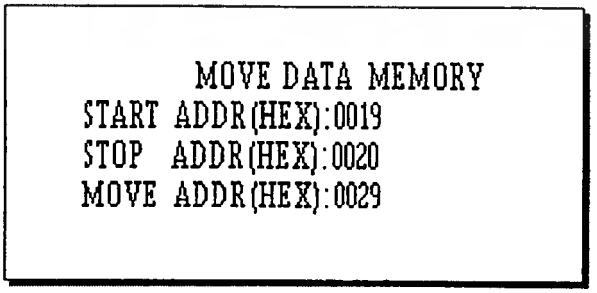


รูป 5.14 ทำการเขียนข้อมูลลงบน Data Memory

ทำการอ่านข้อมูลที่ได้ทำการเขียนลงไปโดยการเลือกข้อ 1 แอดเดรสที่ต้องการอ่านคือ 07EFH ซึ่งข้อมูลที่ได้ก็คือ FEH ตรงกับที่ได้เขียนลงไปตอนแรก

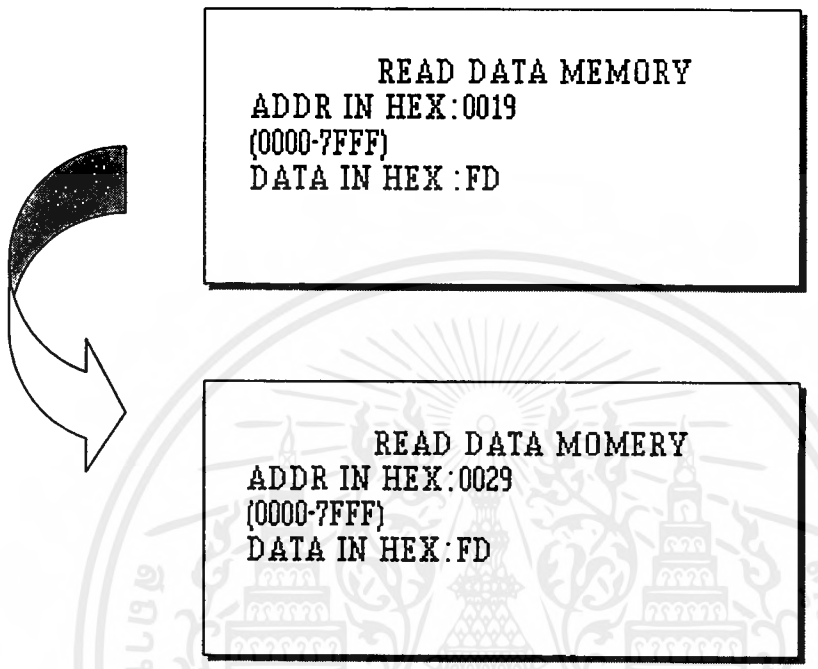
รูป 5.15 ทำการอ่านข้อมูลจาก Data Memory

ต่อไปลองทำการคัดลอกข้อมูลโดยเลือกข้อ 3 MOVE ซึ่งจะทดลองคัดลอกข้อมูล ตั้งแต่แอดเดรส 0019H ถึง แอดเดรส 0020H ไปไว้ที่แอดเดรส 0029H ดังรูปที่ 5.16



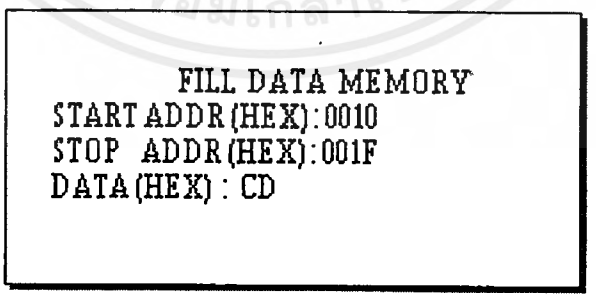
รูป 5.16 แสดงการคัดลอกข้อมูลโดยใช้คำสั่ง MOVE

จากนั้นทำการเช็คข้อมูลที่ทำการคัดลอกโดยการอ่านข้อมูลที่แอดเดรส 0019H เปรียบเทียบกับแอดเดรส 0029H ซึ่งผลที่ได้ปรากฏเท่ากันดังรูปที่ 5.17



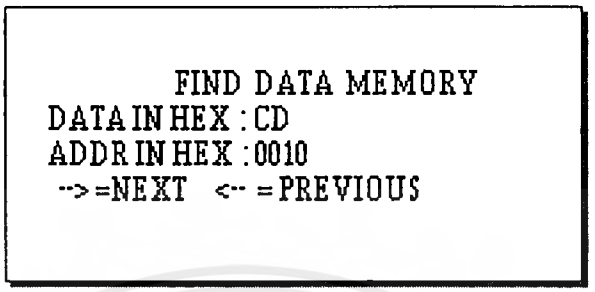
รูป 5.17 แสดงการเปรียบเทียบข้อมูลที่ทำการคัดลอก

การใส่ข้อมูลโดยการเติมข้อมูลค่าเดียวกันลงบนช่วงแอดเดรสที่ต้องการ โดยทำการทดลอง ใส่ค่า 0CDH ลงบนแอดเดรส 0010H ถึง 001FH โดยการใช้คำสั่ง FILL <ข้อ 4 > ดังรูปที่ 5.18



รูป 5.18 แสดงการเติมข้อมูลโดยใช้ คำสั่ง FILL

จากนั้นการตรวจสอบโดยการใช้คำสั่ง FIND <ข้อ 5 > ซึ่งจะทำการหาข้อมูล 0CDH โดยเริ่มที่ แอดเดรส 0010H ดังรูป 5.19

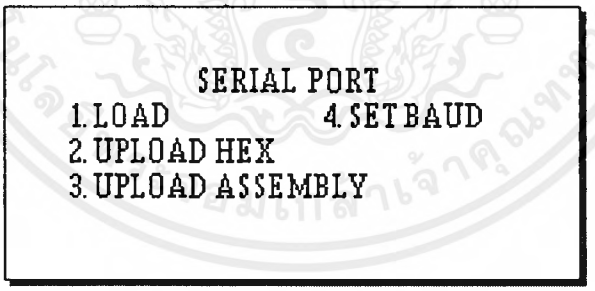


รูป 5.19 แสดงการหาข้อมูลโดยใช้คำสั่ง FIND

ทำการกดปุ่ม →เพื่อดูข้อมูลแอดเดรสถัดไป ซึ่งพบว่าจะพบข้อมูลที่มีค่า CD ทุก ๆ แอดเดรสตั้งแต่ 0010H-001FH ตรงตามการทดลองขั้นต้น

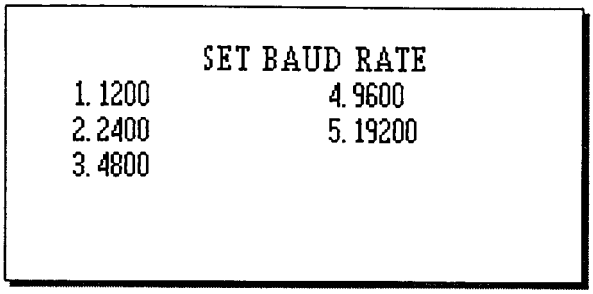
5.4 การติดต่อกับไมโครคอมพิวเตอร์

จากเมนูหลักเมื่อทำการเลือกข้อ 4 SERIAL ซึ่งจะเป็นคำสั่งที่ใช้ในการติดต่อกับพีซีดังรูปที่ 5.20



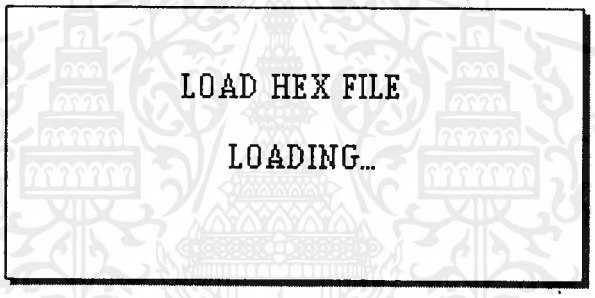
รูป 5.20 เมนูย่อยของคำสั่งที่ใช้ในการติดต่อกับพี่ซี

ขั้นแรกทำการตั้งค่าความเร็วในการส่งข้อมูลทางพอร์ทอนุกรม (Serial Port) โดยการเลือกข้อ3 ซึ่งจะมีค่าให้เลือกดังรูปที่ 5.21



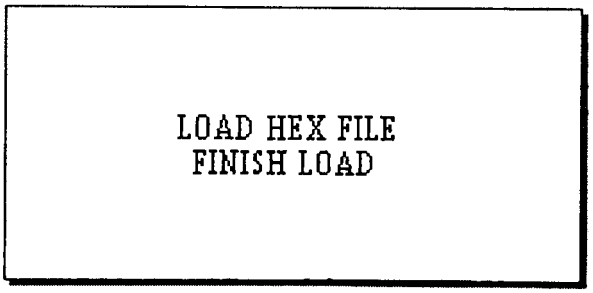
รูป 5.21 การตั้งค่าความเร็วในการติดต่อทางพอร์ทอนุกรม

จากนั้นทดลองดาว์นโหลดโปรแกรมที่อยู่ในรูป HEX Code จาก พีซี มายังแอดเดรส 8000H เป็นต้นไป โดยการเลือกข้อ1 บอร์ดจะรอข้อมูลจาก พีซีดังรูปที่ 5.22



รูป 5.22 การรันโปรแกรมทางพอร์ทขนานจากพีซี

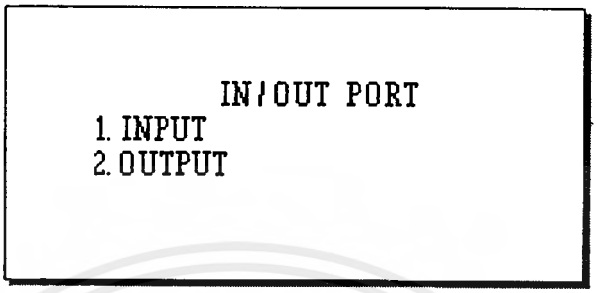
จากนั้นก็ทำการคัดลอก HEX File บนพีซีลงบนบอร์ดผ่านทางพอร์ทอนุกรม COM1 โดยการพิมพ์ copy tests.hex com1 เมื่อบอร์ดได้รับโปรแกรมจากพีซีจนครบแล้วจะแสดงสถานะดังรูปที่ 5.23 แล้วกลับไป เมนูหลัก



รูป 5.23 แสดงสถานะของการรับโปรแกรมจากพีซีผ่านทางพอร์ทขนาน

5.5 การส่งข้อมูลออกทางพอร์ทของผู้ใช้ (USER PORT)

โดยการเลือก ข้อ 4 จาก เมนเมนู จะเป็นการส่งข้อมูลออกทางพอร์ทของผู้ใช้ ซึ่งมีเมนูย่อย ดังแสดงในรูป 5.24

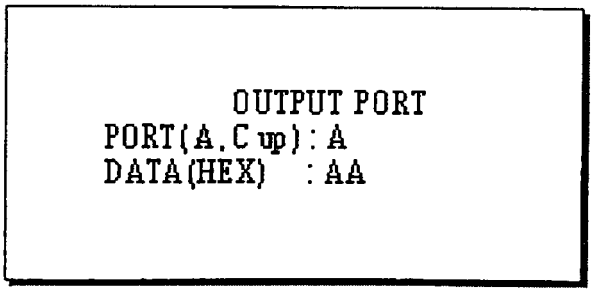


รูป 5.24 เมนูย่อยของการส่งข้อมุลออกจากพอร์ทของผู้ใช้

เมื่อทำการเลือกข้อ 1 จากเมนูย่อย จะเป็นการรับข้อมูลจากอินพุทพอร์ทของผู้ใช้ คือ พอร์ท B และ พอร์ท C 4บิตล่าง ดังรูป 5.25 เป็นการรับค่าผ่านทางพอร์ท B ซึ่งจะแสดงค่า ของข้อมูลที่ได้รับทันที

รูป 5.25 การรับข้อมูลจากพอร์ทของผู้ใช้

เมื่อต้องการส่งข้อมูลออกทางพอร์ทของผู้ใช้สามารถทำได้โดยการเลือกข้อ 2 ซึ่งจะส่งผ่านข้อมูลออกทางพอร์ท A และ C 4 บิตบน ดังรูปที่ 5.26

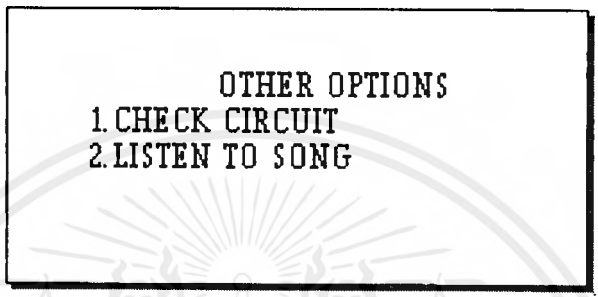


รูป 5.26 การส่งข้อมูลออกทางพอร์ทของผู้ใช้

เมื่อทดลองวัดศักดาที่พอร์ท A ของ 8255 ของผู้ใช้ ปรากฏว่าได้ค่าศักดา 5 V สลับกับ 0 V ตามค่าที่ได้ส่งออกไป

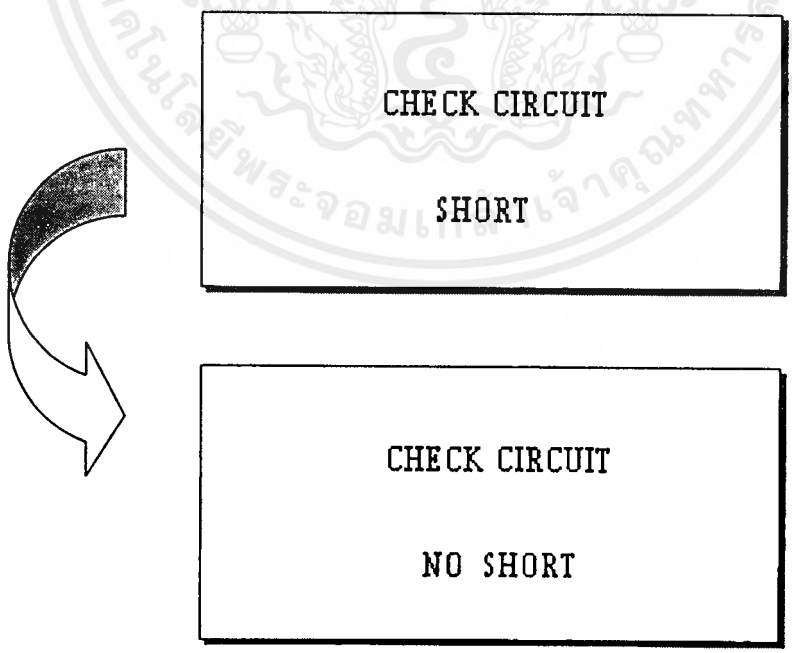
5.6 เครื่องมืออื่น ๆ

เมื่อทำการเลือกข้อ 5 จาก เมนเมนู จะเป็นการเลือกใช้เครื่องมืออื่น ๆ ที่มีอยู่บนบอร์ด ซึ่งมีเมนูย่อย ดังรูป 5.27



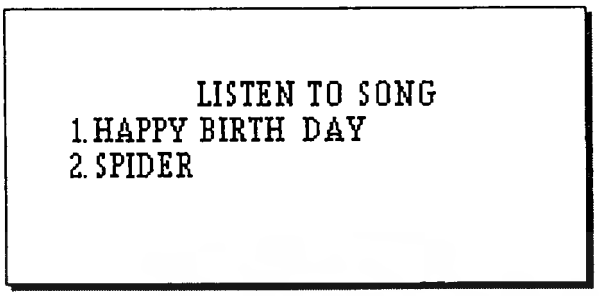
รูป 5.27 เมนูย่อย ของเครื่องมือต่าง ๆ

5.6.1 อุปกรณ์เช็คการเชื่อมต่อ โดยการเลือกข้อ 1 เป็นการเช็คสถานะของการเชื่อมต่อผ่านทางพอร์ทเครื่องมือ ดังรูป ที่ 5.28



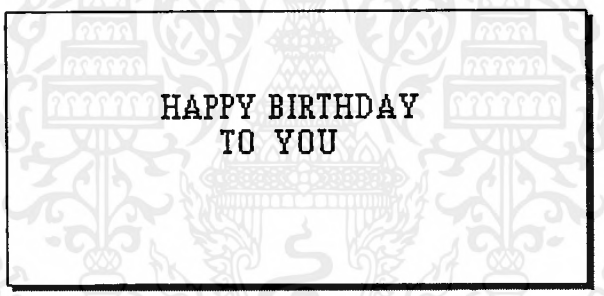
รูป 5.28 แสดงสถานะการเชื่อมต่อผ่านทางพอร์ทเครื่องมือ

5.6.2 ดนตรี โดยการเลือกข้อ 2 จะมีดนตรีให้เลือกฟัง 2 เพลงดังรูปที่ 5.29



รูปที่ 5.29 เมนูเลือกเล่นดนตรี

หากเลือกเพลงที่ 1 จะมีเสียงดนตรีและมี ข้อความดังรูปที่ 5.30



รูปที่ 5.30 เพลง Happy Birth Day

บทที่ 6
สรุปและวิจารณ์โครงงาน

6.1 สรุปผลการปฏิบัติงาน

ในการปฏิบัติงานครั้งนี้ได้ผลเป็นที่น่าพอใจเป็นอย่างยิ่ง เพราะสามารถทำงานได้บรรลุตามเป้าหมาย ซึ่งก็คือ ซิงเกิลบอร์ดที่ใช้งานได้เป็นอย่างดี สามารถเขียนโปรแกรมเป็นภาษาแอสแซมบลีลงไปได้ทันที และทำงานตามโปรแกรมที่เขียนลงไปได้ มีการเช็คคำสั่งภาษาแอสแซมบลีที่พิมพ์ลงไปบรรทัดต่อบรรทัดสามารถ ตรวจสอบความถูกต้องได้ทันที ทำให้ง่ายต่อการหาจุดผิดพลาดของโปรแกรมที่ทำการเขียนได้อย่างรวดเร็ว ฟังก์ชั่นอื่นๆ ที่จะช่วยอำนวยความสะดวกสามารถทำงานได้เป็นอย่างดี ทำให้สะดวกในการติดต่อกับอุปกรณ์ภายในเอง เช่น การเช็คข้อมูลที่อยู่ในหน่วยความจำและรีจิสเตอร์ต่างๆ ทำให้การเช็คการทำงานของโปรแกรมทำได้สะดวกมากยิ่งขึ้น การติดต่อสื่อสารกับไมโครคอมพิวเตอร์ทำได้อย่างถูกต้อง ข้อมูลที่ทำการส่งถึงกันไม่มีความผิดพลาด

ดังนั้นซิงเกิลบอร์ดนี้จึงสามารถที่จะทำงานได้ด้วยตัวของมันเอง โดยไม่ต้องอาศัยไมโครคอมพิวเตอร์ อีกต่อไปซึ่งก็จะทำให้การนำไปประยุกต์ ใช้งานทำได้สะดวก และมีความยืดหยุ่นสูง ซึ่งจะเป็นประโยชน์ต่อนักพัฒนาโปรแกรมที่ไม่มีไมโครคอมพิวเตอร์ไว้ใช้งาน และช่วยทำให้ไมโครคอนโทรลเลอร์ มีประสิทธิภาพในการทำงานที่กว้างขวางมากยิ่งขึ้น

6.2 ปัญหาและอุปสรรคในการทำงาน

ในการจัดทำโครงสร้างงานทางด้านฮาร์ดแวร์พบกับอุปสรรคมากพอสมควร ปัญหาที่พบในการทำงานมีดังนี้

1.การออกแบบระบบทั้งหมดต้องกระทำอย่างรอบคอบ เพราะการติดต่อกับอุปกรณ์ต่าง ๆ ต้องมีการอ้างถึงแอดเดรสของอุปกรณ์นั้น ๆ ซึ่งบางครั้งวงจรในส่วนที่ออกแบบไว้แล้วบางครั้งต้องกลับมาทำใหม่ เพราะอุปกรณ์ต่าง ๆ อาจมีตำแหน่งแอดเดรสทับกันอยู่ ซึ่งจะทำให้ระบบผิดพลาด เช่น วงจรถอดรหัส หน่วยความจำข้อมูล ตำแหน่งแอดเดรส 8000H ขึ้นไปมีแอดเดรสซ้ำกับ I/O

2.การเลือกใช้อุปกรณ์บนวงจรต้องกระทำให้ประหยัดที่สุดเพราะหากทำการออกแบบโดยไม่คำนึงถึงในส่วนนี้อาจทำให้บอร์ดมีขนาดใหญ่มากเกินไป เช่น ไอซีเบอร์ 74LS04 ซึ่งเป็นอินเวอร์เตอร์ภายในไอซี 1 ตัว จะมีทั้งหมด 6 เกท ซึ่งการออกแบบควรใช้อุปกรณ์ในส่วนนี้ ให้มากที่สุด แทนที่จะใช้เกทตัวอื่นๆ เพิ่มขึ้นมาอีก 1 เบอร์ ซึ่งจะเป็นการเปลืองเนื้อที่บนบอร์ด

3.การออกแบบแผ่นลายวงจรพิมพ์กระทำได้ยากลำบาก ต้องใช้ความระวังเป็นพิเศษ เนื่องจากอุปกรณ์ทั้งระบบมีจำนวนมาก การเดินลายทองแดงต้องกระทำอย่างรอบคอบ ถึงจะใช้แผ่นวงจรแบบ 2 หน้าก็ตาม ซึ่งเนื่องจากมีลายทองแดงเป็นจำนวนมาก ดังนั้นลายทองแดงแต่ละเส้นจึงมีขนาดเล็กมาก และอยู่ใกล้กันมาก ซึ่งอาจทำให้เกิดการช๊อตได้

4.เนื่องจากซอฟท์แวร์ที่ใช้ในการออกแบบแผ่นลายวงจรพิมพ์ นั้นยังไม่เป็นที่นิยม จึงเกิดความไม่สะดวก ในการหาร้านที่จะทำแผ่นลายวงจรพิมพ์ได้ละเอียด เพราะ ลายวงจรที่ออกแบบไว้มีความละเอียดสูงพอสมควร

5.เนื่องจากอุปกรณ์หลายตัวที่ใช้งานไม่มีในไลบรารีของซอฟท์แวร์ที่ใช้ในการออกแบบลายวงจรพิมพ์ จึงต้องทำการสร้างไลบรารีขึ้นมาเอง ซึ่งเมื่อได้แผ่นวงจรลายพิมพ์ที่เสร็จสมบูรณ์มาแล้วพบว่าอุปกรณ์บางตัว มีขนาดของขา ไม่ตรงกับช่องเสียบของแผ่นลายวงจรที่ออกแบบมา

6.อุปกรณ์ต่างๆ เมื่อนำลงบนแผ่นลายวงจรพิมพ์แล้ว ในขั้นแรกไม่สามารถทำงานได้ เพราะลายวงที่ได้ออกแบบไว้ มีความผิดพลาด จึงทำให้เสียเวลามาก กว่าอุปกรณ์ทางฮาร์ดแวร์ ทั้งหมดจะทำงานได้ดังที่ออกแบบไว้

7.ในการออกแบบส่วนของซอร์ฟแวร์ เริ่มต้นได้ช้าและยาก เพราะขาดแนวทางในการออกแบบอัลกอริทึม ต้องใช้เวลาในการศึกษานานพอสมควร ในการเริ่มต้น

8.รีจิสเตอร์ที่มีอยู่ภายในตัวไมโครคอนโทรลเลอร์ไม่เพียงพอต่อการใช้งาน เพราะต้องมีการเก็บข้อมูลเดิมไว้มากในขณะที่มีการทำโปรแกรมย่อย เพื่อคงข้อมูลเดิมที่สำคัญไว้สำหรับการกลับ มาทำงานที่โปรแกรมหลัก

9.การเขียนโปรแกรมที่ต้องมีการแก้ไขอยู่ตลอดเวลา การใช้อีพรอมธรรมดาในการเก็บตัวโปรแกรม มีความไม่สะดวกเป็นอย่างยิ่ง เพราะต้องคอยล้างข้อมูลและเขียนข้อมูล อยู่ตลอดเวลทำให้การดำเนินการแก้ไข เสียเวลาค่อนข้างมาก

6.3 แนวทางในการแก้ไข

1.ในการออกแบบวงจรถอดรหัสแอดเดรสของวงจรส่วนต่าง ๆ ควรจะคิดให้รอบคอบว่าทั้งระบบนั้นต้องการส่วนประกอบอะไรบ้าง จากนั้นจึงค่อยทำการออกแบบวงจรถอดรหัส เพื่อที่จะได้ไม่ต้องทำในส่วนของวงจรถอดรหัสบ่อย ๆ และเป็นการหลีกเลี่ยง ไม่ให้อุปกรณ์ ต่าง มีแอดเดรสทับกันอีกด้วย

2.ในการเลือกใช้อุปกรณ์ ไอซีตัวหนึ่ง ๆ ควรจะถูกใช้ส่วนประกอบข้างในให้หมด เช่นไอซีเบอร์ 74LS00 ซึ่งเป็นแนนด์เกท ภายในจะมีแนนด์เกททั้งหมด 4ตัว ดังนั้นในส่วนของการออกแบบจึง

ใช้ความรู้ทางดิจิตอลลอจิก เปลี่ยนอุปกรณ์อย่างอื่นที่อยู่ในเกทอื่น ให้อยู่ในรูปของแนนด์เกท เพื่อที่จะทำการลดอุปกรณ์บนบอร์ดลงไป ทำให้บอร์ดมีขนาดเล็กลง และเป็นการประหยัดอีกด้วย

3.การออกแบบแผ่นสายวงจรเพื่อลดความยุ่งยาก ควรจะใช้ความสามารถของซอฟท์แวร์ที่ใช้ในเรื่องของการเดินสายโดยอัตโนมัติ (Auto Route) ซึ่งก็จะทำให้ไม่ต้องมานั่งลากลายวงจรเอง ซึ่งถ้าหากวงจรมีความละเอียดและมีขนาดใหญ่ ซึ่งถ้าใช้วิธีการเดินสายแบบอัตโนมัติ จะช่วยลดความยุ่งยากลงไปได้มาก แล้วโอกาสผิดพลาดยังมีน้อยอีกด้วย

4.ในส่วนของการหาร้านที่รับทำแผ่นลายลายวงจรพิมพ์ ควรจะเลือกร้านที่สามารถทำแผ่นลายวงจรได้ละเอียด และ ชิ้นงานมีความเรียบร้อยหากมีปัญหาในเรื่องซอฟท์แวร์ที่ใช้เวลาจะนำแผ่นลายวงจรพิมพ์ไปสั่งทำก็ควรจะนำซอฟท์แวร์ที่ใช้ไปด้วยเลย ซึ่งจะช่วยตัดปัญหาในเรื่องของซอฟท์แวร์ที่ทางร้านอาจจะไม่มีออกไปได้

5.การออกแบบแผ่นลายวงจรพิมพ์ หากอุปกรณ์ที่ใช้งานไม่มีในไลบรารี ของซอฟท์แวร์ การเขียนไลบรารีขึ้นมาเองจะต้องกระทำอย่างละเอียดที่สุดเมื่อทำเสร็จแล้ว ควรที่จะพิมพ์ออกมาเพื่อลองวางอุปกรณ์จริงดูก่อน ซึ่งจะได้ไม่มีปัญหาในภายหลัง เมื่อทำแผ่นลายวงจรพิมพ์ออกมาแล้ว

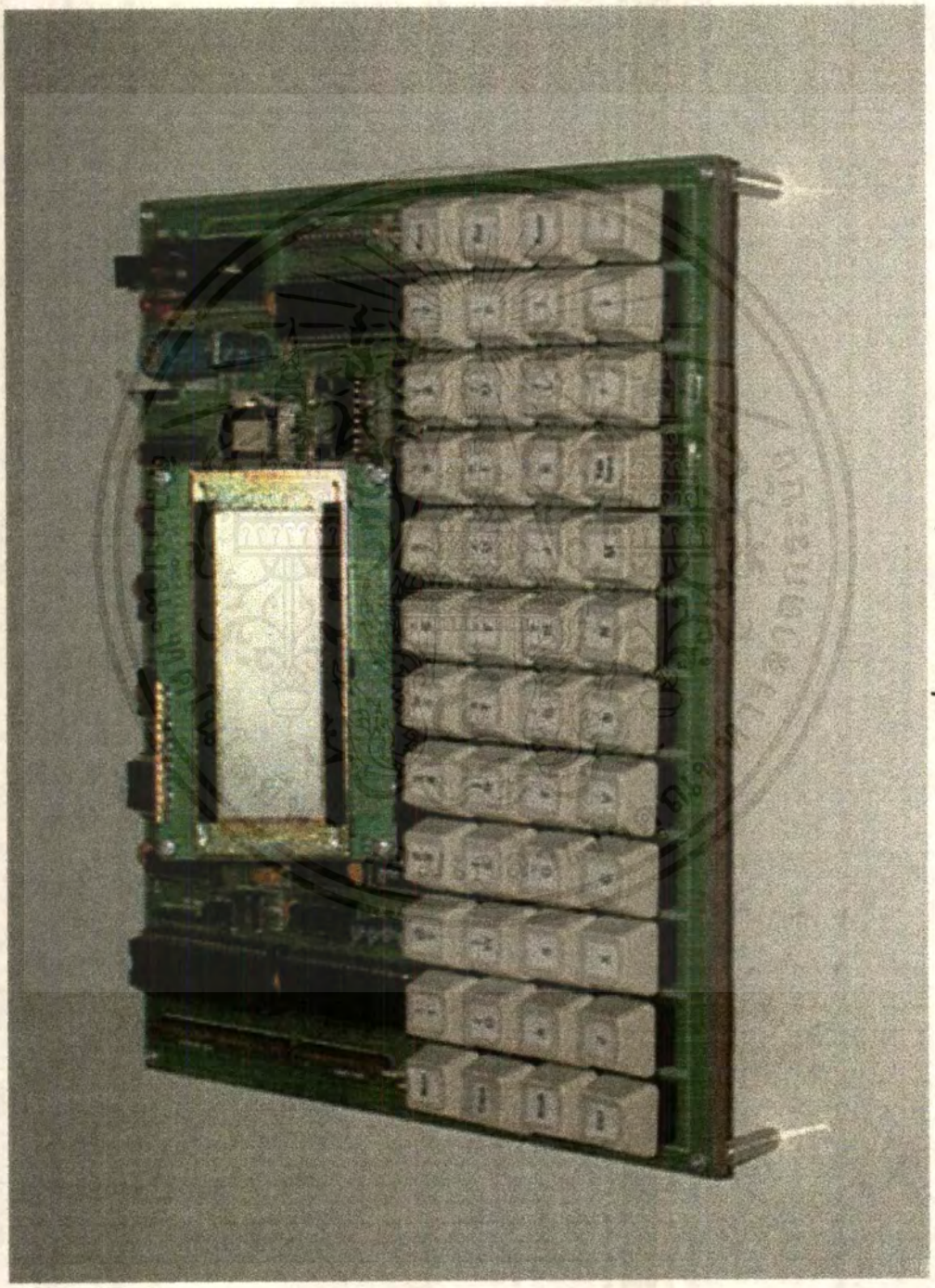
6.ศึกษาวิธีการเขียนโปรแกรม จากซิงเกิลบอร์ดสำเร็จรูปทั่วไป ที่มีอยู่ตามท้องตลาดว่ามีอัลกอริทึมอย่างไรในการออกแบบ แล้วทำการออกแบบโปรแกรม เพื่อใช้เป็นแนวทางในการออกแบบ ทำให้ง่ายต่อการเริ่มต้น ที่จะทำการเขียนโปรแกรม

7.การใช้งานรีจิสเตอร์ภายในที่มีอยู่อย่างจำกัด ควรใช้อย่างคุ้มค่าที่สุด คือเก็บแต่ข้อมูลที่มีความสำคัญเท่านั้นในรีจิสเตอร์ ข้อมูลที่สำคัญน้อยกว่าก็ควร หลีกเลี่ยงโดยการนำไปเก็บบนหน่วยความจำภายในแทน

8.หลีกเลี่ยงการใช้อีพรอมในขั้นตอนของการออกแบบโปรแกรม ควรใช้อุปกรณ์สำเร็จรูปอย่างอื่น เช่นอีพรอมอีมูเลเตอร์แทน ทำให้สะดวกในการแก้ไขโปรแกรม เป็นอย่างมาก

6.4 สรุป

ซิงเกิลบอร์ดที่ได้ทำการออกแบบสามารถทำงานได้อย่างถูกต้อง แต่ก็ยังมีโปรแกรมส่วนอื่นๆที่ควรมีเพิ่มเติม เช่นการรันโปรแกรมทีละขั้นตอน (Single Step) วงจรที่ใช้ในการสร้างระบบเวลาจริง ซึ่งงานในส่วนนี้ ยังไม่ได้ดำเนินงานเนื่องจากมีเวลาไม่พอ แต่ตัวซิงเกิลบอร์ดในตอนนี้สามารถนำไปใช้งานได้อย่างมีประสิทธิภาพ มีความสะดวกในการนำไปใช้งานเพราะมีความยืดหยุ่นในการทำงานสูง



รูปที่ 6.1 ซิงเกิลบอร์ด

ภาคผนวก

ภาคผนวก ก

ชุดคำสั่งภายใน MCS-51

ตารางที่ ก.1 ตารางแสดงชุดคำสั่งทั้งหมดภายใน MCS-51 (8051 Instruction Set)

Mnemonic	Description-Arithmetic Operations	Byte	Osc. Period
1. ARITHMETIC OPERATIONS			
ADD A,Rn	Add register to Accumulator	1	12
ADD A,direct	Add direct byte to Accumulator	2	12
ADD A,@Ri	Add indirect RAM to Accumulator	1	12
ADD A,#data	Add immediate data to Accumulator	2	12
ADDC A,Rn	Add register to Acc. with Carry	1	12
ADDC A,direct	Add direct byte to Acc. with Carry	2	12
ADDC A,@Ri	Add indirect RAM to Acc. with Carry	1	12
ADDC A,#data	Add immediate data to Acc. / Carry	2	12
SUBB A,Rn	Subtract reg. from Acc. with borrow	1	12
SUBB A,direct	Sub. direct byte from Acc. / borrow	2	12
SUBB A,@Ri	Sub. indirect RAM from Acc./ borrow	1	12
SUBB A,#data	Sub. imm. data from Acc. / borrow	2	12
INC A	Increment Accumulator	1	12
INC Rn	Increment register	1	12
INC direct	Increment direct byte	2	12
INC @Ri	Increment indirect RAM	1	12
DEC A	Decrement Accumulator	1	12
DEC Rn	Decrement register	1	12
DEC direct	Decrement direct byte	2	12
DEC @Ri	Decrement indirect RAM	1	12
INC DPTR	Increment Data Pointer	1	24

Mnemonic	Description-Arithmetic Operations	Byte	Osc. Period
MUL AB	Multiply A and B	1	48
DIV AB	Divide A by B	1	48
DA A	Decimal adjust Accumulator	1	12
2. LOGICAL OPERATIONS			
ANL A,Rn	AND register to Accumulator	1	12
ANL A,direct	AND direct byte to Accumulator	2	12
ANL A,@Ri	AND indirect RAM to Accumulator	1	12
ANL A,#data	AND immediate data to Accumulator	2	12
ANL direct,A	AND Accumulator to direct byte	2	12
ANL direct,#data	AND immediate data to direct byte	3	24
ORL A,Rn	OR register to Accumulator	1	12
ORL A,direct	OR direct byte to Accumulator	2	12
ORL A,@Ri	OR indirect RAM to Accumulator	1	12
ORL A,#data	OR immediate data to Accumulator	2	12
ORL direct,A	OR Accumulator to direct byte	2	12
ORL direct,#data	OR immediate data to direct byte	3	24
XRL A,Rn	Exc-OR register to Accumulator	1	12
XRL A,direct	Exc-OR direct byte to Accumulator	2	24
XRL A,@Ri	Exc-OR indirect RAM to Accumulator	1	12
XRL A,#data	Exc-OR immediate data to Acc.	2	12
XRL direct,A	Exc-OR Accumulator to direct byte	2	12
XRL direct,#data	Exc-OR imm: data to direct byte	3	24
CLR A	Clear Accumulator	1	12
CPL A	Complement Accumulator	1	12
RL A	Rotate Accumulator left	1	12
RLC A	Rotate Acc. left through Carry	1	12

Mnemonic	Description-Arithmetic Operations	Byte	Osc. Period
RR A	Rotate Accumulator right	1	12
RRC A	Rotate Acc. right through Carry	1	12
SWAP A	Swap nibbles within the Accumulator	1	12
3. DATA TRANFER			
MOV A,Rn	Move register to Accumulator	1	12
MOV A,direct	Move direct byte to Accumulator	2	12
MOV A,@Ri	Move indirect RAM to Accumulator	1	12
MOV A,#data	Move immediate data to Accumulator	2	12
MOV Rn,A	Move Accumulator to register	1	12
MOV Rn,direct	Move direct byte to register	2	24
MOV Rn,#data	Move immediate data to register	2	12
MOV direct,A	Move Accumulator to direct byte	2	12
MOV direct,Rn	Move register to direct byte	2	24
MOV direct,direct	Move direct byte to direct byte	3	24
MOV direct,@Ri	Move indirect RAM to direct byte	2	24
MOV direct,#data	Move immediate data to direct byte	3	24
MOV @Ri,A	Move Accumulator to indirect RAM	1	12
MOV @Ri,direct	Move direct byte to indirect RAM	2	24
MOV @Ri,#data	Move immediate data to indirect RAM	2	12
MOV DPTR,#data16	Load Data Pointer with 16-bit const	3	24
MOVC A,@A+DPTR	Move Code byte rel. to DPTR to Acc.	1	24
MOVC A,@A+PC	Move Code byte rel. to PC to Acc.	1	24
MOVX A,@Ri	Move Ext. RAM (8-bit addr.) to Acc.	1	24
MOVX A,@DPTR	Move Ext. RAM (16-bit addr) to Acc.	1	24
MOVX @Ri,A	Move Acc. to Ext. RAM (8-bit addr.)	1	24
MOVX @DPTR,A	Move Acc. to Ext. RAM (16-bit addr)	1	24

Mnemonic	Description-Arithmetic Operations	Byte	Osc. Period
PUSH direct	Push direct byte onto stack	2	24
POP direct	Pop direct byte from stack	2	24
XCH A,Rn	Exchange register with Accumulator	1	12
XCH A,direct	Exchange direct byte with Acc.	2	12
XCH A,@Ri	Exchange indirect RAM with Acc.	1	12
XCHD A,@Ri	Exchange low order digit indirect RAM with Accumulator	1	12
4. BOOLEAN VARIABLE MANIPULATION			
CLR C	Clear Carry	1	12
CLR bit	Clear direct bit	2	12
SETB C	Set Carry	1	12
SETB bit	Set direct bit	2	12
CPL C	Complement Carry	1	12
CPL bit	Complement direct bit	2	12
ANL C,bit	AND direct bit to Carry	2	24
ANL C,/bit	AND complement of dir. bit to Carry	2	24
ORL C,bit	OR direct bit to Carry	2	24
ORL C,/bit	OR complement of dir. bit to Carry	2	24
MOV C,bit	Move direct bit to Carry	2	12
MOV bit,C	Move Carry to direct bit	2	24
5. PROGRAM AND MACHINE CONTROL			
JC rel	Jump if Carry is set	2	24
JNC rel	Jump if Carry not set	2	24

Mnemonic	Description-Arithmetic Operations	Byte	Osc. Period
JB bit,rel	Jump if direct bit is set	3	24
JNB bit,rel	Jump if direct bit is not set	3	24
JBC bit,rel	Jump if dir. bit is set & clear bit	3	24
ACALL addr11	Absolute subroutine call	2	24
LCALL addr16	Long subroutine call	3	24
RET	Return from subroutine	1	24
RETI	Return from interrupt	1	24
AJMP addr11	Absolute jump	2	24
LJMP addr16	Long jump	3	24
SJMP rel	Short jump (relative address)	2	24
JMP @A+DPTR	Jump indirect relative to the DPTR	1	24
JZ rel	Jump if Accumulator is zero	2	24
JNZ rel	Jump if Accumulator is not zero	2	24
CJNE A,direct,rel	Compare direct byte to Accumulator and jump if not equal	3	24
CJNE A,#data,rel	Compare immediate data to Accumulator and jump if not equal	3	24
CJNE Rn,#data,rel	Compare immediate data to register and jump if not equal	3	24
CJNE @Ri,#data,rel	Compare immediate data to indirect RAM and jump if not equal	3	24
DJNZ Rn,rel	Decr. register and jump if not zero	2	24
DJNZ direct,rel	Decrement direct byte and jump if not zero	3	24
NOP	No operation	1	12

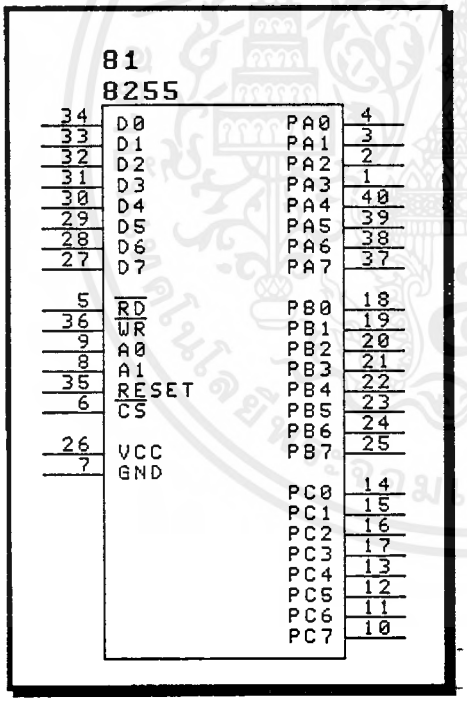
Notes on Instruction Set and Addressing Modes

Rn	=	Register R0 - R7 of the currently selected register bank.
direct	=	8-bit internal data location's address. This could be an internal Data RAM location (0-127) or a SFR.
@Ri	=	8-bit internal Data RAM location addressed indirectly through register R0 or R1.
#data	=	8-bit constant included in instruction.
#data16	=	16-bit constant included in instruction.
addr11	=	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K byte page of Program Memory as the first byte of the following instruction.
addr16	=	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64K byte Program Memory address space.
rel	=	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
bit	=	Direct addressed bit in internal Data RAM or SFR.

ภาคผนวก ข

การใช้งาน 8255

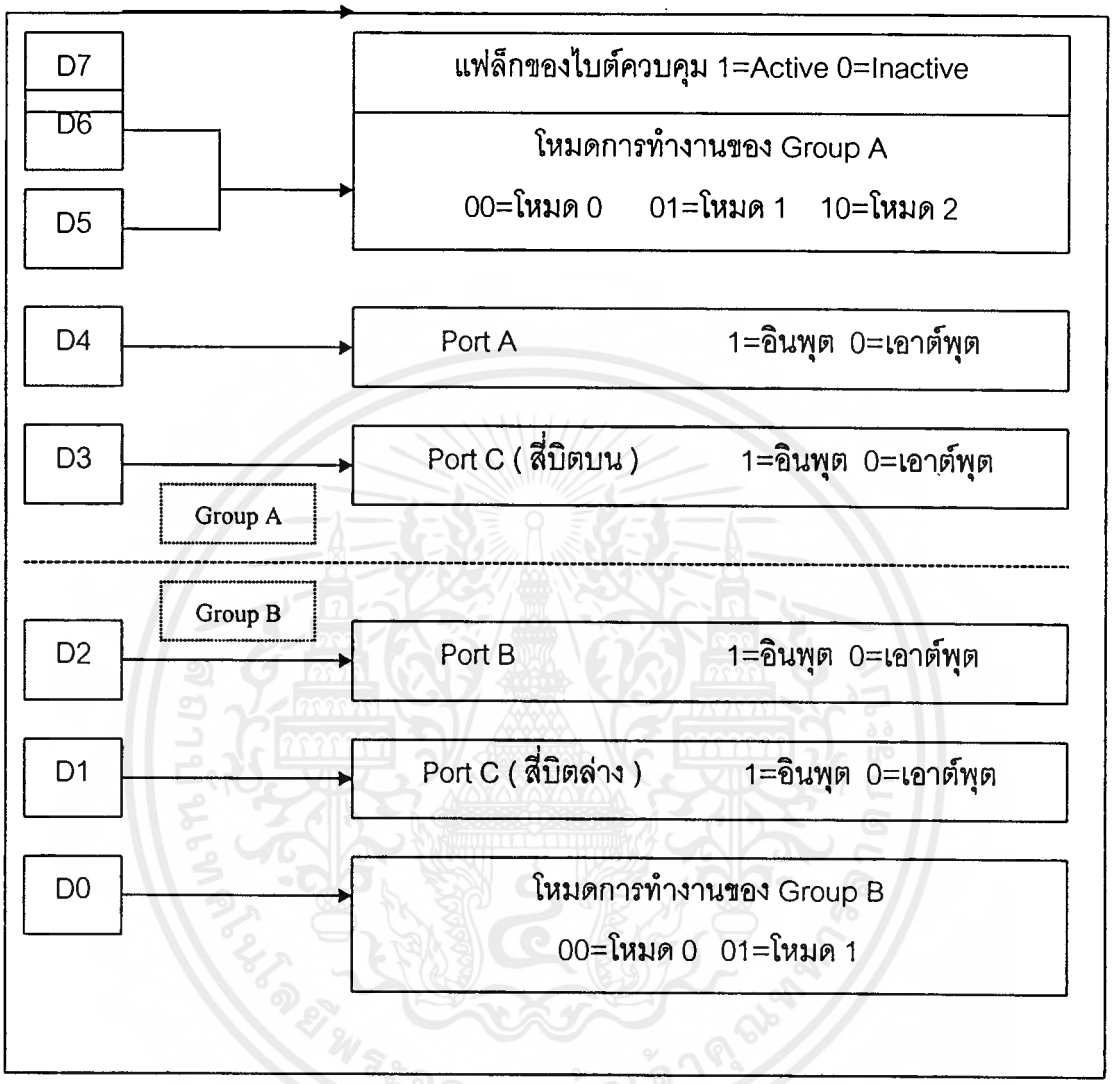
8255 เป็นไอซีที่มี 40 ขา ได้รับการออกแบบมาให้มีสัญญาณเพื่อเชื่อมต่อกับ 8080 โดยสัญญาณนี้เหมาะที่จะใช้กับ MCS-51 ได้เช่นเดียวกัน โดยไอซีดังกล่าวถูกออกแบบมาเพื่อใช้เป็นพอร์ทอินพุท-เอาต์พุทขยายให้กับไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์ได้ถึง 3 พอร์ทด้วยกัน โดยขึ้นกับการส่งข้อมูลควบคุมดังในรูปที่ข.2 โดยมีตำแหน่งของขาสัญญาณดังในรูปที่ ข.1



การเรียกพอร์ทของ 8255 จะเรียกพอร์ทต่างๆว่า พอร์ท A, พอร์ท B และพอร์ท C โดยพอร์ท C แยกเป็น 2 ส่วนคือ พอร์ท C ล่าง หรือตั้งแต่ PC0-PC3 มีจำนวน 4 บิต และพอร์ท C บน หรือตั้งแต่ PC4-PC7 ที่พิเศษคือ ทุกพอร์ทเป็นได้ทั้งพอร์ทอินพุท และพอร์ทเอาต์พุท

รูปที่ ข.1

ภาพแสดงตำแหน่งการวางขาของ 8255



รูปที่ ข.2

ภาพแสดงความหมายของบิตข้อมูลควบคุมสำหรับ 8255

โดยในการอ่าน-เขียน พอร์ทใดๆรวมทั้งการเขียนคำสั่งควบคุม จะต้องมีการกำหนดสัญญาณ $\overline{RD}$, $\overline{WR}$, $\overline{A0}$, $\overline{A1}$ ดังในตารางที่ ข.1

ตารางที่ ข.1 แสดงสัญญาณควบคุมการทำงานของ 8255

$\overline{RD}$	$\overline{WR}$	$\overline{A0}$	$\overline{A1}$	ความหมาย
1	0	0	0	เขียนพอร์ท A ซึ่งเป็นข้อมูล
0	1	0	0	อ่านพอร์ท A ซึ่งเป็นข้อมูล
1	0	0	1	เขียนพอร์ท B ซึ่งเป็นข้อมูล
0	1	0	1	อ่านพอร์ท B ซึ่งเป็นข้อมูล
1	0	1	0	เขียนพอร์ท C ซึ่งเป็นข้อมูล
0	1	1	0	อ่านพอร์ท C ซึ่งเป็นข้อมูล
1	0	1	1	เขียนข้อมูล ซึ่งเป็นรหัสควบคุม
0	1	1	1	ไม่มีความหมายใดๆ

ภาคผนวก ค
การใช้งาน LCD

LCD (Liquid Crystal Display) เป็นอุปกรณ์แสดงผลแบบหนึ่งที่นิยมใช้มาก และพบเห็นกันบ่อยๆ LCD สามารถแบ่งประเภทตามลักษณะของการแสดงผลได้ 3 แบบคือ LCDแบบอักขระ (Character LCD Module), แบบกราฟิก (Graphic LCD Module) และ LCD แบบเซกเมนต์ (Segment LCD Module)

LCD แบบอักขระ เป็นโมดูล LCD ที่สามารถแสดงตัวอักษร,ตัวเลข,และเครื่องหมายต่างๆได้ โดยสร้างจากจุดเล็ก ๆ หรือเรียกว่า ดอตเมตริกซ์ ซึ่งก็จะมีขนาดความกว้างและสูงของอักขระแต่ละตัว โดยทั่วๆไปมี 2 ขนาดคือ 5x7 จุด 5x10 จุด นอกจากนั้น LCD แบบนี้สามารถแสดงข้อความได้ 1 บรรทัด หรือมากกว่าก็ได้ ขึ้นอยู่กับรุ่นของ LCD นั้นๆ

ขาสัญญาณของโมดูล LCD แบบอักขระ

โมดูล LCD แบบอักขระที่นำมาเป็นตัวอย่างนี้เป็นแบบ 1 บรรทัด 16 ตัวอักษร มีขาต่อใช้งานทั้งสิ้น 14 ขา ได้แก่

Vss (ขา 1): ต่อกราวนด์

Vdd (ขา 2): ต่อไฟเลี้ยง +5 โวลท์

Vo (ขา 3): เป็นขาอินพุทสำหรับป้อนแรงดัน เพื่อปรับความเข้มของการแสดงผล

RS (ขา 4): เป็นขาอินพุท ใช้เลือกว่าข้อมูลที่ทำการส่งในขณะนั้นเป็นข้อมูลคำสั่งสำหรับรีจิสเตอร์คำสั่งของ LCD หรือเป็นข้อมูลสำหรับรีจิสเตอร์ข้อมูลแสดงผลของ LCD โดยถ้าขานี้เป็น"0" ข้อมูลที่ส่งมาจะเป็นข้อมูลคำสั่ง แต่ถ้าเป็น "1" ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

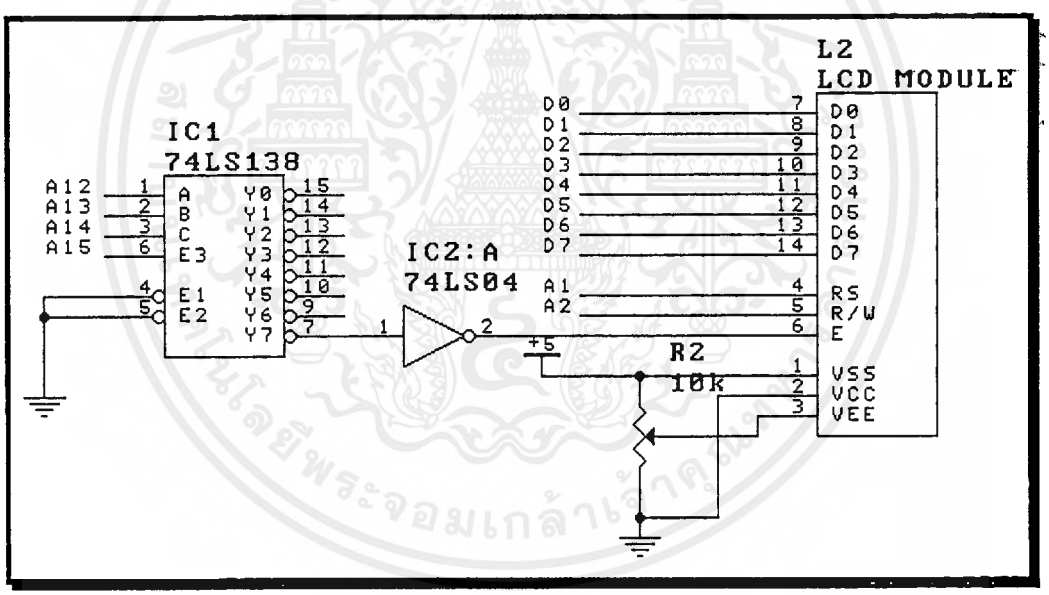
$R/\overline{W}$ (ขา 5): เป็นขาที่ใช้เลือกว่าจะอ่านหรือเขียนข้อมูลกับ LCD ถ้าหากเป็น "0" จะเป็นการเขียนข้อมูล แต่ถ้าเป็น "1" จะเป็นการอ่านข้อมูล

E (ขา 6): เป็นขา Enable ให้ LCD ทำงาน

DB0-DB7 (ขา 7-14): เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต

ตารางที่ ค.1 แสดงความสัมพันธ์ของสัญญาณ E, RS, และ R/$\overline{W}$ ของ LCD

RS	R/$\overline{W}$	E	การทำงาน
0	0	↘	เขียนคำสั่ง
0	1	⌒	อ่านสถานะของ LCD
0	0	↘	เขียนข้อมูล
0	1	⌒	อ่านข้อมูล



รูปที่ ค.1 แสดงตัวอย่างการเชื่อมต่อ LCD

ตารางที่ ค.2 คำสั่งควบคุม LCD

Instruction	RS	R W	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0	Description
Clear display	0	0	0	0	0	0	0	0	0	1	Clear all display and returns the cursor to the home position
Return home	0	0	0	0	0	0	0	0	1	x	Return the cursor to the home position.DD RAM contents remain unchanged
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor ON/OFF (C), blink of cursor position character (B)
Cursor and display shift	0	0	0	0	0	1	S/C	R/ L	x	x	Moves the cursor and shifts the display without changing DD RAM contents

Instruction	RS	R W	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0	Description
Function set	0	0	0	0	1	DL	N	F	x	x	Sets interface data length (DL), number of display lines (L), character font (F)
Set CG RAM address	0	0	0	1	A_{CG}						Sets the CG RAM address.
Set DD RAM address	0	0	1	A_{DD}							Sets the DD RAM address
Read busy flag & address	0	1	BF	AC							Read Busy flag (BF) indicating internal operation is being performed and reads address counter contents
Write data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM
Read data to CG or DD RAM	1	1	Read Data								Reads data into DD RAM or CG RAM

ภาคผนวก ง

การใช้งาน MAX232

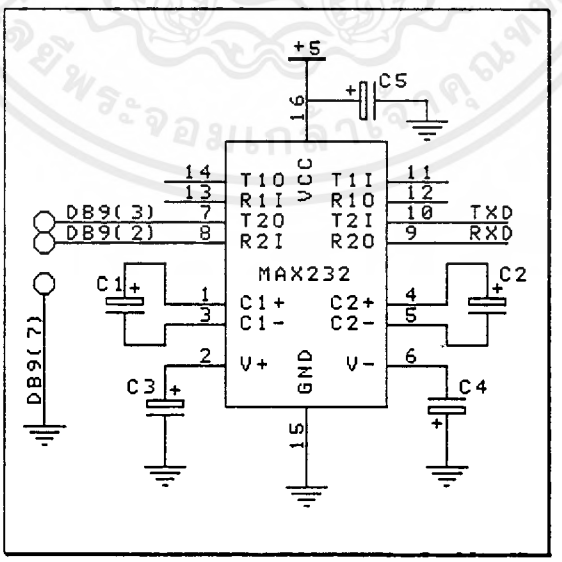
ในการเชื่อมต่อแบบอนุกรมเข้ากับอุปกรณ์คอมพิวเตอร์ต่างๆ มักจะกำหนดใช้การเชื่อมต่อตามมาตรฐาน RS-232 Cทั้งนี้เพื่อให้มีการใช้งานเส้นสัญญาณหรือรูปแบบของตัวเชื่อม ที่สอดคล้องกัน

แต่เนื่องจากระดับแรงดันที่ใช้และการแทนความหมายของระดับลอจิกตามมาตรฐานนี้แตกต่างไปจากที่ใช้งานกันในระบบดิจิตอลทั่วไป โดยระดับสัญญาณของ RS-232C เป็นแบบไบโพลาร์ (Bipolar) กล่าวคือ

- ระดับแรงดัน -3 V ถึง -20 V แทนค่าลอจิก 1
- ระดับแรงดัน +3 V ถึง +20 V แทนค่าลอจิก 0

จะเห็นได้ว่ามีความจำเป็นต้องเพิ่มเติมอุปกรณ์หรือวงจรพิเศษเข้าไปเพื่อเปลี่ยนระดับแรงดันจากระบบ 0 ถึง +5 V จากขาสัญญาณของ 8051 ให้เป็นไปตามมาตรฐาน RS-232C

โดยทั่วไปรูปแบบของวงจรสามารถทำได้ในหลายลักษณะ ทั้งการออกแบบสร้างวงจรโดยการใช้ทรานซิสเตอร์ หรือใช้ไอซีวงจรรวมที่เรียกว่า ไดรเวอร์ ได้แก่เบอร์ MAX232 ซึ่งมีตัวอย่างการใช้งานดังรูปที่ ง.1



ภาพที่ ง.1 แสดงการใช้งาน MAX232

ในการใช้งาน MAX232 ให้เชื่อมต่อตัวเก็บประจุที่ขา C1, C2, ระหว่าง V- กับ GND, และระหว่าง V+ กับ GND ดังรูป ง.1 โดยค่าของตัวเก็บประจุที่ใช้ควรจะมีค่าประมาณ 10 uF สำหรับ MAX232 และ 100 nF สำหรับ MAX232A

ตารางที่ ง.1 แสดงหน้าที่ของขาสัญญาณต่างๆใน MAX232

pin	name function
1	c1+ \
3	c1- > +5v to +10v voltage doubler
2	V+ (10V) /
4	c2+ \
5	c2- > +10V to -10V voltage inverter
6	V- (-10V) /
15	GND
16	VCC (5V)
11	T1in \
14	T1out \ TTL/CMOS inputs to
10	T2in / RS-232 outputs
7	T2out /
12	R1out \
13	R1in \ RS-232 inputs to
9	R2out / TTL/CMOS outputs
8	R2in /

ภาคผนวก จ

Program Monitor

```
;********** SINGLE BOARD **********
;********** VERSION 1.0 **********
;***** HARDWARE 8031 BOARD *****
;***** SOFTWARE ENG. PEERACHAT *****

;********** VARIABLE SET **********
S8255A EQU 0FF00H
S8255B EQU 0FF01H
S8255C EQU 0FF02H
S8255D EQU 0FF03H

U8255A EQU 0FF20H
U8255B EQU 0FF21H
U8255C EQU 0FF22H
U8255D EQU 0FF23H

KBD EQU 0FF40H

LCDWRC EQU 0FF60H    ;R/W=0,RS=0
LCDWRD EQU 0FF61H    ;R/W=0,RS=1
LCDRDC EQU 0FF62H    ;R/W=1,RS=0
LCDRDD EQU 0FF63H    ;R/W=1,RS=1

BRAT12 EQU 0E8H
BRAT24 EQU 0F4H
BRAT48 EQU 0FAH
BRAT96 EQU 0FDH

SMOD EQU B.7

;********** INTERNAL RAM **********
    ORG 0000H
    ;***** USER AREA *****
    DS 8        ;USER AREA (REGISTER BANK 0)
    DS 24       ;USER AREA (INTERNAL RAM & USER STACK)
    DS 15       ;USER AREA (BIT ADDRESSABLE)
    DS 1        ;SYSTEM FLAG AREA

    ;***** BUFFER AREA *****
ASMBUF:  DS 8
FINDBUF: DS 1
INPBUF:  DS 20

    ;***** MEMORY AREA *****
STTADD: DS 2
ENDADD: DS 2
MEMADD: DS 2
DESADD: DS 2
RUNADD: DS 2
SCAADD: DS 2
TABMEM: DS 4
POWMEM: DS 4

    ;***** GENERAL AREA *****
IXDATA: DS 2
IYDATA: DS 2
IZDATA: DS 2
STACK:  DS 24
ENDINT:
;SYSTEM FLAG (EQU) *****
SCANOL  EQU 78H
SHIFTF  EQU 79H
CAPF    EQU 7AH
CTRLF   EQU 7BH
STMF    EQU 7CH
SENDEN  EQU 7DH
POWF    EQU 7EH
ERRF    EQU 7FH

;********** PROGRAM ***********
    ORG 0000H
    LJMP RES

    ORG 0003H
    ;LJMP EXTINT0      ;INT0

    ORG 000BH
    ;LJMP RTCINT       ;TF0

    ORG 0013H
    ;LJMP EXTINT1      ;INT1

    ORG 001BH
    ;LJMP T1INT        ;TF1

    ORG 0023H
    ;LJMP SERINT       ;TI+RI

    ;********** RESET **********
    ORG 0030H
RES:  CLR A
    MOV PSW,A         ;SELECT REGISTER BANK 0
    MOV R7,A          ;POWER UP DELAY
RES1:  MOV R6,A
    DJNZ R6,$
    DJNZ R7,RES1
    LCALL PBEEP
    MOV SP,#STACK
    CLR EA

;********** INITIAL 8255 **********
    MOV A,#83H        ;A,C UPPER=OUTPUT
                      ;B,C LOWER=INPUT
    MOV DPTR,#S8255D
    MOVX @DPTR,A
    MOV DPTR,#U8255D
    MOVX @DPTR,A

    MOV A,#0FFH       ;ALL OUPUT PORT HIGH
    MOV DPTR,#S8255A
    MOVX @DPTR,A
    MOV DPTR,#S8255C
    MOVX @DPTR,A
    MOV DPTR,#U8255A
    MOVX @DPTR,A
    MOV DPTR,#U8255C
    MOVX @DPTR,A

;********** INITIAL RTC **********
```

```
;********** SET DEFAULT **********
    MOV TMOD,#20H       ;TIMER 1 MODE 2
    MOV SCON,#52H       ;SERIAL 8 BIT UART MODE
    MOV TH1,#BRAT96     ;9600
    SETB TR1
    MOV IE,#0E4H
    MOV IP,#0E0H
    MOV P1,#0FFH

    JNB POWF,RES7       ;CHECK PASSWORD
RES2:  MOV DPTR,#PASST1
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
RES3:  MOV R0,#4
    MOV A,#9CH
    LCALL LCDWI
RES4:  LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC RES4
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC RES5
    CJNE A,#41H,$+3     ;< A JUMP
    JC RES4
    CJNE A,#5BH,$+3     ;> Z JUMP
    JC RES5
    CJNE A,#61H,$+3     ;< a JUMP
    JC RES4
    CJNE A,#7BH,$+3     ;> z JUMP
    JNC RES4
RES5:  LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,RES4

    MOV A,#9CH
    LCALL LCDWI
    LCALL LCDRD
    CJNE A,POWMEM,RES6
    LCALL LCDRD
    CJNE A,POWMEM+1,RES6
    LCALL LCDRD
    CJNE A,POWMEM+2,RES6
    LCALL LCDRD
    CJNE A,POWMEM+3,RES6
    SJMP RES7

RES6:  MOV DPTR,#PASST2
    LCALL LCDOUT3
    LCALL LBEEP
    MOV R2,#10
    LCALL DTSEC
    SJMP RES2

RES7:  MOV R0,#08H      ;CLEAR INTERNAL RAM (ALL)
    MOV R1,#ENDINT-08H
RES8:  MOV @R0,#00H
    INC R0
    DJNZ R1,RES8
    LCALL LCDSET
    MOV DPTR,#TITLE
    LCALL LCDOUT
    LCALL HBEEP
    MOV R2,#20
    LCALL DTSEC
    MOV RUNADD,#80H
    MOV RUNADD+1,#00H
    LJMP MAIN

PASST1: DB '  ENTER PASSWORD  '
    DB '                '
    DB '                '
    DB '                '
PASST2: DB ' INCORRECT PASSWORD '

TITLE:  DB '  SINGLE BOARD  '
    DB '  VERSION  1.0  '
    DB 'BY  Yong,Moo and Ton'
    DB 'ELECTRONIC  ENGINEER'

MAIN:  MOV SP,#STACK
    MOV DPTR,#MAINT0
    LCALL LCDOUT
MAIN0:  LCALL SCAN

    CJNE A,#31H,$+9     ;PROGRAM
    LCALL BLACK1
    LJMP MAIN1

    CJNE A,#32H,$+9     ;MEM. & REG.
    LCALL BLACK2
    LJMP MAIN2

    CJNE A,#33H,$+9     ;SERIAL
    LCALL BLACK3
    LJMP MAIN3

    CJNE A,#34H,$+9     ;IN/OUT
    LCALL BLACK4
    LJMP MAIN4

    CJNE A,#35H,$+9     ;OTHERS
    LCALL BLACK5
    LJMP MAIN5

    CJNE A,#36H,MAIN0   ;QUIT
    LCALL BLACK6
    LJMP QUIT

MAINT0: DB '   MAIN MENU   '
    DB '1.PROGRAM 4.IN/OUT '
    DB '2.MEM&REG 5.OTHERS '
    DB '3.SERIAL  6.QUIT   '

MAIN1:  MOV DPTR,#MAINT1
    LCALL LCDOUT
MAIN10: LCALL SCAN

    CJNE A,#31H,$+9     ;ASSEMBLER
    LCALL BLACK1
    LJMP ASM

    CJNE A,#32H,$+9     ;UNASSEMBLER
    LCALL BLACK2
```

```
    LJMP UASM

    CJNE A,#33H,MAIN10   ;RUN PROGRAM
    LCALL BLACK3
    LJMP RUN

MAINT1: DB '   PROGRAM    '
    DB '1.ASSEMBLER     '
    DB '2.UNASSEMBLER   '
    DB '3.RUN PROGRAM   '

MAIN2:  MOV DPTR,#MAINT2
    LCALL LCDOUT
MAIN20: LCALL SCAN

    CJNE A,#31H,S+9      ;PROGRAM MEMORY
    LCALL BLACK1
    LJMP RDPRG

    CJNE A,#32H,S+9      ;DATA MEMORY
    LCALL BLACK2
    LJMP MAIN22

    CJNE A,#33H,S+9      ;INTERNAL RAM
    LCALL BLACK3
    LJMP MAIN23

    CJNE A,#34H,MAIN20   ;REGISTER
    LCALL BLACK4
    LJMP MAIN24

MAINT2: DB ' MEMORY & REGISTER '
    DB '1.PROGRAM 4.SFR  '
    DB '2.DT MEM.        '
    DB '3.INT RAM        '

MAIN22:  MOV DPTR,#MAINT22
    LCALL LCDOUT
MAIN220: LCALL SCAN

    CJNE A,#31H,S+9      ;READ DATA MEMORY
    LCALL BLACK1
    LJMP RDDT

    CJNE A,#32H,S+9      ;WRITE DATA MEMORY
    LCALL BLACK2
    LJMP WDDT

    CJNE A,#33H,S+9      ;MOVE DATA MEMORY
    LCALL BLACK3
    LJMP MOVDT

    CJNE A,#34H,S+9      ;FILL DATA MEMORY
    LCALL BLACK4
    LJMP FLLDT

    CJNE A,#35H,MAIN220  ;FIND DATA MEMORY
    LCALL BLACK5
    LJMP FNDDT

MAINT22: DB '   DATA MEMORY   '
    DB '1.READ   4.FILL  '
    DB '2.WRITE  5.FIND  '
    DB '3.MOVE           '

MAIN23:  MOV DPTR,#MAINT23
    LCALL LCDOUT
MAIN230: LCALL SCAN

    CJNE A,#31H,S+9      ;READ INTERNAL RAM
    LCALL BLACK1
    LJMP RDIN

    CJNE A,#32H,S+9      ;WRITE INTERNAL RAM
    LCALL BLACK2
    LJMP WDIN

    CJNE A,#33H,MAIN230  ;FIND INTERNAL RAM
    LCALL BLACK3
    LJMP FNDIN

MAINT23: DB '   INTERNAL RAM   '
    DB '1.READ           '
    DB '2.WRITE          '
    DB '3.FIND           '

MAIN24:  MOV DPTR,#MAINT24
    LCALL LCDOUT
MAIN240: LCALL SCAN

    CJNE A,#31H,S+9      ;READ SFR
    LCALL BLACK2
    LJMP RDSFR

    CJNE A,#32H,MAIN240  ;WRITE SFR
    LCALL BLACK3
    LJMP WDSFR

MAINT24: DB ' SPECIAL FUNCTION '
    DB '    REGISTER     '
    DB '1.READ           '
    DB '2.WRITE          '

MAIN3:  MOV DPTR,#MAINT3
    LCALL LCDOUT
MAIN30: LCALL SCAN

    CJNE A,#31H,S+9      ;LOAD PROGRAM
    LCALL BLACK1
    LJMP LOAD

    CJNE A,#32H,S+9      ;UPLOAD HEX PROGRAM
    LCALL BLACK2
    LJMP UPLD

    CJNE A,#33H,S+9      ;UPLOAD ASSEMBY PROGRAM
    LCALL BLACK3
    LJMP UPAS

    CJNE A,#34H,MAIN30   ;SET BAUD RATE
    LCALL BLACK4
    LJMP BAUD

MAINT3: DB '   SERIAL PORT   '
```

```
    DB '1.LOAD   4.SET BAUD'
    DB '2.UPLOAD HEX     '
    DB '3.UPLOAD ASSEMBLY '

MAIN4:  MOV DPTR,#MAINT4
    LCALL LCDOUT
MAIN40: LCALL SCAN

    CJNE A,#31H,$+9      ;INPUT
    LCALL BLACK1
    LJMP INPUT

    CJNE A,#32H,MAIN40    ;OUTPUT
    LCALL BLACK2
    LJMP OUTPUT

MAINT4: DB '  IN/OUT PORT   '
    DB '1.INPUT          '
    DB '2.OUTPUT         '
    DB '                 '

MAIN5:  MOV DPTR,#MAINT5
    LCALL LCDOUT
MAIN50: LCALL SCAN

    CJNE A,#31H,$+9      ;CHECK CIRCUIT
    LCALL BLACK1
    LJMP CHKCC

    CJNE A,#32H,MAIN50    ;LISTEN TO SONG
    LCALL BLACK2
    LJMP PSONG

MAINT5: DB '  OTHER OPTIONS  '
    DB '1.CHECK CIRCUIT    '
    DB '2.LISTEN TO SONG   '
    DB '                   '

MAIN51:  MOV DPTR,#MAINT51
    LCALL LCDOUT
    MOV R2,#20
    LCALL DTSEC
    LJMP MAIN

MAINT51: DB '                '
    DB '     TIMER       '
    DB '                 '
    DB '                 '

;********** SUB FOR FUTURE **********

;********** UNASSEMBLER **********
UASM:  MOV DPTR,#UASMT
    LCALL LCDOUT
    MOV R2,#10
    LCALL DTSEC
    MOV STTADD,#0A0H
    MOV STTADD+1,#00H
    MOV MEMADD,#80H
    MOV MEMADD+1,#00H
    MOV DPH,STTADD
    MOV DPL,STTADD+1
    MOV A,#38H
    MOVX @DPTR,A
    INC DPTR
    MOV A,#30H
    MOVX @DPTR,A
    INC DPTR
    MOVX @DPTR,A
    INC DPTR
    MOVX @DPTR,A
    LJMP ASM3
UASMT:  DB '                 '
    DB '   UASSEMBLER    '
    DB '                 '
    DB '                 '

;********** RUN USER PROGRAM **********
RUN:   MOV DPTR,#RUNT1
    LCALL LCDOUT
    MOV DPTR,#0A000H
    MOV A,#02H
    MOVX @DPTR,A
    INC DPTR
    MOV A,#80H
    MOVX @DPTR,A
    INC DPTR
    MOV A,#00H
    MOVX @DPTR,A

    MOV DPH,RUNADD
    MOV DPL,RUNADD+1
    CLR A
    JMP @A+DPTR
RUNT1:  DB '                 '
    DB '   RUN PROGRAM    '
    DB '                 '
    DB '                 '

;********** READ DATA IN PROGRAM MEMORY **********
RDPRG:  MOV DPTR,#RDPRGT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
RDPRG1: MOV R0,#4
    MOV A,#0CDH
    LCALL LCDWI
RDPRG2: LCALL SCAN
    CJNE A,#30H,$+3      ;< 0 JUMP
    JC RDPRG2
    CJNE A,#3AH,$+3      ;> 9 JUMP
    JC RDPRG3
    CJNE A,#41H,$+3      ;< A JUMP
    JC RDPRG2
    CJNE A,#47H,$+3      ;> F JUMP
    JC RDPRG3
    CJNE A,#61H,$+3      ;< a JUMP
    JC RDPRG2
    CJNE A,#67H,$+3      ;> f JUMP
    JNC RDPRG2
RDPRG3: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,RDPRG2
```

```
    MOV A,#0CDH         ;READ ADDRESS FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV DPH,A
    CJNE A,#80H,$+3     ;CHECK ADDRESS
    JC RDPRGER
    CJNE A,#0FCH,$+3
    JNC RDPRGER
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV DPL,A
    MOVX A,@DPTR

    MOV R1,A            ;WRITE DATA TO LCD
    MOV A,#0E1H
    LCALL LCDWI
    MOV A,R1
    LCALL HTOA
    MOV A,R2
    LCALL LCDWD
    MOV A,R3
    LCALL LCDWD
    SJMP RDPRGI

RDPRGER: MOV DPTR,#ADDERT   ;ADDRESS ERROR
    LCALL LCDOUT
    LCALL EBEEP
    MOV R2,#20
    LCALL DTSEC
    LJMP RDPRG

RDPRGT: DB 'READ PROGRAM MEMORY '
    DB 'ADDR IN HEX:    '
    DB '(8000-9FFF)     '
    DB 'DATA IN HEX:    '
ADDERT: DB '            '
    DB ' ADDRESS ERROR  '
    DB '                '
    DB '                '

;********** READ DATA IN DATA MEMORY **********
RDDT:  MOV DPTR,#RDDTT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
RDDT1: MOV R0,#4
    MOV A,#0CDH
    LCALL LCDWI
RDDT2: LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC RDDT2
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC RDDT3
    CJNE A,#41H,$+3     ;< A JUMP
    JC RDDT2
    CJNE A,#47H,$+3     ;> F JUMP
    JC RDDT3
    CJNE A,#61H,$+3     ;< a JUMP
    JC RDDT2
    CJNE A,#67H,$+3     ;> f JUMP
    JNC RDDT2
RDDT3: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,RDDT2

    MOV A,#0CDH         ;READ ADDRESS FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV DPH,A
    CJNE A,#80H,$+3     ;CHECK ADDRESS
    JNC RDDTER
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV DPL,A
    MOVX A,@DPTR

    MOV R1,A            ;WRITE DATA TO LCD
    MOV A,#0E1H
    LCALL LCDWI
    MOV A,R1
    LCALL HTOA
    MOV A,R2
    LCALL LCDWD
    MOV A,R3
    LCALL LCDWD
    SJMP RDDTI

RDDTER: MOV DPTR,#ADDERT    ;ADDRESS ERROR
    LCALL LCDOUT
    LCALL EBEEP
    MOV R2,#20
    LCALL DTSEC
    LJMP RDDT

RDDTT: DB ' READ DATA MEMORY '
    DB 'ADDR IN HEX:    '
    DB '(0000-7FFF)     '
    DB 'DATA IN HEX:    '

;********** WRITE DATA IN DATA MEMORY **********
WDDT:  MOV DPTR,#WDDTT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
WDDT1: MOV R0,#4
    MOV A,#0CDH
    LCALL LCDWI
WDDT2: LCALL SCAN
```

```
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC WDDT2
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC WDDT3
    CJNE A,#41H,$+3     ;< A JUMP
    JC WDDT2
    CJNE A,#47H,$+3     ;> F JUMP
    JC WDDT3
    CJNE A,#61H,$+3     ;< a JUMP
    JC WDDT2
    CJNE A,#67H,$+3     ;> f JUMP
    JNC WDDT2
WDDT3: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,WDDT2

    MOV A,#0CDH         ;READ ADDRESS FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV MEMADD,A
    CJNE A,#80H,$+3     ;CHECK ADDRESS
    JNC WDDTER
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV MEMADD+1,A

WDDT4:  MOV R0,#2
    MOV A,#0E1H
    LCALL LCDWI
WDDT5:  LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC WDDT5
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC WDDT6
    CJNE A,#41H,$+3     ;< A JUMP
    JC WDDT5
    CJNE A,#47H,$+3     ;> F JUMP
    JC WDDT6
    CJNE A,#61H,$+3     ;< a JUMP
    JC WDDT5
    CJNE A,#67H,$+3     ;> f JUMP
    JNC WDDT5
WDDT6:  LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,WDDT5

    MOV A,#0E1H         ;READ DATA FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV DPH,MEMADD
    MOV DPL,MEMADD+1
    MOVX @DPTR,A        ;WRITE DATA TO DATA MEMORY
    LJMP WDDT1

WDDTER: MOV DPTR,#ADDERT    ;ADDRESS ERROR
    LCALL LCDOUT
    LCALL EBEEP
    MOV R2,#20
    LCALL DTSEC
    LJMP WDDT

WDDTT:  DB ' WRITE DATA MEMORY '
    DB 'ADDR IN HEX:    '
    DB '(0000-7FFF)     '
    DB 'DATA IN HEX:    '

;********** MOVE DATA IN DATA MEMORY **********
MOVDT:  MOV DPTR,#MOVDTT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
MOVDT1: MOV R0,#4
    MOV A,#0D0H
    LCALL LCDWI
MOVDT2: LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC MOVDT2
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC MOVDT3
    CJNE A,#41H,$+3     ;< A JUMP
    JC MOVDT2
    CJNE A,#47H,$+3     ;> F JUMP
    JC MOVDT3
    CJNE A,#61H,$+3     ;< a JUMP
    JC MOVDT2
    CJNE A,#67H,$+3     ;> f JUMP
    JNC MOVDT2
MOVDT3: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,MOVDT2

    MOV A,#0D0H         ;READ START ADDRESS FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV STTADD,A
    CJNE A,#80H,$+3     ;CHECK ADDRESS
    JC $+5
    LJMP MOVDTER
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV STTADD+1,A

MOVDT4: MOV R0,#4
```

```
    MOV A,#0A4H
    LCALL LCDWI
MOVDT5: LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC MOVDT5
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC MOVDT6
    CJNE A,#41H,$+3     ;< A JUMP
    JC MOVDT5
    CJNE A,#47H,$+3     ;> F JUMP
    JC MOVDT6
    CJNE A,#61H,$+3     ;< a JUMP
    JC MOVDT5
    CJNE A,#67H,$+3     ;> f JUMP
    JNC MOVDT5
MOVDT6: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,MOVDT5

    MOV A,#0A4H         ;READ END ADDRESS FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV ENDADD,A
    CJNE A,#80H,$+3     ;CHECK ADDRESS
    JC $+5
    LJMP MOVDTER
    MOV DPH,STTADD
    CJNE A,DPH,$+3
    JNC $+5
    LJMP MOVDTER
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV ENDADD+1,A
    MOV DPL,STTADD+1
    CJNE A,DPL,$+3
    JNC $+5
    LJMP MOVDTER

MOVDT7: MOV R0,#4
    MOV A,#0E4H
    LCALL LCDWI
MOVDT8: LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC MOVDT8
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC MOVDT9
    CJNE A,#41H,$+3     ;< A JUMP
    JC MOVDT8
    CJNE A,#47H,$+3     ;> F JUMP
    JC MOVDT9
    CJNE A,#61H,$+3     ;< a JUMP
    JC MOVDT8
    CJNE A,#67H,$+3     ;> f JUMP
    JNC MOVDT8
MOVDT9: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,MOVDT8

    MOV A,#0E4H         ;READ MOVE ADDRESS FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV DESADD,A
    CJNE A,#80H,$+3     ;CHECK ADDRESS
    JNC MOVDTER
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV DESADD+1,A

MOVDT10: MOV DPH,STTADD     ;MOVE DATA
    MOV DPL,STTADD+1
    MOVX A,@DPTR
    MOV DPH,DESADD
    MOV DPL,DESADD+1
    MOVX @DPTR,A
    INC DPTR
    MOV DESADD,DPH
    MOV DESADD+1,DPL
    MOV A,DPH
    CJNE A,#80H,$+3
    JNC MOVDTF
    MOV DPH,STTADD
    MOV DPL,STTADD+1
    INC DPTR
    MOV STTADD,DPH
    MOV STTADD+1,DPL
    MOV A,ENDADD
    CJNE A,DPH,$+3
    JNC $+5
    LJMP MOVDT1
    MOV A,ENDADD+1
    CJNE A,DPL,$+3
    JNC $+5
    LJMP MOVDT1
    SJMP MOVDT10

MOVDTER: MOV DPTR,#ADDERT   ;ADDRESS ERROR
    LCALL LCDOUT
    MOV R2,#20
    LCALL DTSEC
    LJMP MOVDT

MOVDTF:  POP DPL
    POP DPH
    MOV DPTR,#MOVDTFT   ;ADDRESS FULL
    LCALL LCDOUT
    LCALL LBEEP
    MOV R2,#20
    LCALL DTSEC
```

```
        LJMP MOVDT

MOVDTT:  DB ' MOVE DATA MEMORY '
        DB 'START ADDR(HEX):  '
        DB 'STOP  ADDR(HEX):  '
        DB 'MOVE  ADDR(HEX):  '
MOVDTFT: DB '                 '
        DB ' DATA MEMORY FULL '
        DB '                  '
        DB '                  '

;********** COMPARE DATA IN DATA MEMORY **********
COMDT:  LJMP MAIN

;********** FILL DATA IN DATA MEMORY **********
FLLDT:  MOV DPTR,#FLLDTT
        LCALL LCDOUT
        MOV A,#0FH
        LCALL LCDWI
FLLDT1: MOV R0,#4
        MOV A,#0D0H
        LCALL LCDWI
FLLDT2: LCALL SCAN
        CJNE A,#30H,$+3     ;< 0 JUMP
        JC FLLDT2
        CJNE A,#3AH,$+3     ;> 9 JUMP
        JC FLLDT3
        CJNE A,#41H,$+3     ;< A JUMP
        JC FLLDT2
        CJNE A,#47H,$+3     ;> F JUMP
        JC FLLDT3
        CJNE A,#61H,$+3     ;< a JUMP
        JC FLLDT2
        CJNE A,#67H,$+3     ;> f JUMP
        JNC FLLDT2
FLLDT3: LCALL LCDWD
        MOV R2,#2
        LCALL DTSEC
        DJNZ R0,FLLDT2

        MOV A,#0D0H         ;READ START ADDRESS FROM LCD
        LCALL LCDWI
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV STTADD,A
        CJNE A,#80H,$+3     ;CHECK ADDRESS
        JC $+5
        LJMP FLLDTER
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV STTADD+1,A

FLLDT4: MOV R0,#4
        MOV A,#0A4H
        LCALL LCDWI
FLLDT5: LCALL SCAN
        CJNE A,#30H,$+3     ;< 0 JUMP
        JC FLLDT5
        CJNE A,#3AH,$+3     ;> 9 JUMP
        JC FLLDT6
        CJNE A,#41H,$+3     ;< A JUMP
        JC FLLDT5
        CJNE A,#47H,$+3     ;> F JUMP
        JC FLLDT6
        CJNE A,#61H,$+3     ;< a JUMP
        JC FLLDT5
        CJNE A,#67H,$+3     ;> f JUMP
        JNC FLLDT5
FLLDT6: LCALL LCDWD
        MOV R2,#2
        LCALL DTSEC
        DJNZ R0,FLLDT5

        MOV A,#0A4H         ;READ END ADDRESS FROM LCD
        LCALL LCDWI
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV ENDADD,A
        CJNE A,#80H,$+3     ;CHECK ADDRESS
        JC $+5
        LJMP FLLDTER
        MOV DPH,STTADD
        CJNE A,DPH,$+3
        JC FLLDTER
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV ENDADD+1,A
        MOV DPL,STTADD+1
        CJNE A,DPL,$+3
        JC FLLDTER
FLLDT7: MOV R0,#2
        MOV A,#0E0H
        LCALL LCDWI
FLLDT8: LCALL SCAN
        CJNE A,#30H,$+3     ;< 0 JUMP
        JC FLLDT8
        CJNE A,#3AH,$+3     ;> 9 JUMP
        JC FLLDT9
        CJNE A,#41H,$+3     ;< A JUMP
        JC FLLDT8
        CJNE A,#47H,$+3     ;> F JUMP
        JC FLLDT9
        CJNE A,#61H,$+3     ;< a JUMP
        JC FLLDT8
        CJNE A,#67H,$+3     ;> f JUMP
        JNC FLLDT8
FLLDT9: LCALL LCDWD
        MOV R2,#2
        LCALL DTSEC
        DJNZ R0,FLLDT8
```

```
    MOV A,#0E0H        ;READ DATA FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH

    MOV R5,A
FLLDT10: MOV DPH,STTADD
    MOV DPL,STTADD+1
    MOVX @DPTR,A       ;WRITE DATA TO DATA MEMORY
    INC DPTR
    MOV STTADD,DPH
    MOV STTADD+1,DPL
    MOV A,ENDADD
    CJNE A,DPH,S+3
    JNC S+5
    LJMP FLLDT1
    MOV A,ENDADD+1
    CJNE A,DPL,S+3
    JNC S+5
    LJMP FLLDT1
    MOV A,R5
    SJMP FLLDT10

FLLDTER: MOV DPTR,#ADDERT   ;ADDRESS ERROR
    LCALL LCDOUT
    LCALL EBEEP
    MOV R2,#20
    LCALL DTSEC
    LJMP FLLDT

FLLDTT:  DB ' FILL DATA MEMORY '
    DB 'START ADDR(HEX):  '
    DB 'STOP  ADDR(HEX):  '
    DB 'DATA (HEX):       '

;********** FIND DATA IN DATA MEMORY **********
FNDDT:  MOV DPTR,#FNDDTT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
FNDDT1: MOV R0,#2
    MOV A,#0CDH
    LCALL LCDWI
FNDDT2: LCALL SCAN

    CJNE A,#7FH,S+10   ;LEFT
    MOV R2,#2
    LCALL DTSEC
    SJMP FNDDT6
    CJNE A,#7EH,S+10   ;RIGHT
    MOV R2,#2
    LCALL DTSEC
    SJMP FNDDT4
    CJNE A,#30H,S+3    ;< 0 JUMP
    JC FNDDT2
    CJNE A,#3AH,S+3    ;> 9 JUMP
    JC FNDDT3
    CJNE A,#41H,S+3    ;< A JUMP
    JC FNDDT2
    CJNE A,#47H,S+3    ;> F JUMP
    JC FNDDT3
    CJNE A,#61H,S+3    ;< a JUMP
    JC FNDDT2
    CJNE A,#67H,S+3    ;> f JUMP
    JNC FNDDT2
FNDDT3: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,FNDDT2

    MOV A,#0CDH        ;READ DATA FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH

    MOV FINDBUF,A
    MOV MEMADD,#0FFH
    MOV MEMADD+1,#0FFH
FNDDT4: MOV DPH,MEMADD
    MOV DPL,MEMADD+1
    INC DPTR
FNDDT5: MOVX A,@DPTR
    MOV R5,A
    MOV MEMADD,DPH
    MOV MEMADD+1,DPL
    INC DPTR
    MOV A,STTADD
    CJNE A,#80H,S+5
    SJMP FNDFN
    MOV A,R5
    CJNE A,FINDBUF,FNDDT5
    SJMP FNDDT7

FNDDT6: MOV DPH,MEMADD
    MOV DPL,MEMADD+1
    LCALL DPDEC
    MOVX A,@DPTR
    MOV R5,A
    MOV MEMADD,DPH
    MOV MEMADD+1,DPL
    LCALL DPDEC
    MOV A,MEMADD
    CJNE A,#0FFH,S+5
    SJMP FNDFN
    MOV A,R5
    CJNE A,FINDBUF,FNDDT6

FNDDT7: MOV A,#0A1H        ;WRITE ADDRESS TO LCD
    LCALL LCDWI
    MOV A,MEMADD
    LCALL HTOA
    MOV A,R2
    LCALL LCDWD
    MOV A,R3
    LCALL LCDWD
    MOV A,MEMADD+1
    LCALL HTOA
    MOV A,R2
```

```
        LCALL LCDWD
        MOV A,R3
        LCALL LCDWD
        LJMP FNDDT1

FNDFN:  MOV DPTR,#FNDFNT
        LCALL LCDOUT
        LCALL LBEEP
        MOV R2,#20
        LCALL DTSEC
        LJMP FNDDT

FNDDTT: DB ' FIND DATA MEMORY '
        DB 'DATA IN HEX:    '
        DB 'ADDR IN HEX:    '
        DB 00100000B
        DB 01111110B
        DB '=NEXT '
        DB 01111111B
        DB '=PREVIOUS '
FNDFNT: DB '                '
        DB ' FINISH ADDRESS '
        DB '                '
        DB '                '

;********** READ DATA IN INTERNAL RAM **********
RDIN:   MOV DPTR,#RDINT
        LCALL LCDOUT
        MOV A,#0FH
        LCALL LCDWI
RDIN1:  MOV R0,#2
        MOV A,#0CDH
        LCALL LCDWI
RDIN2:  LCALL SCAN
        CJNE A,#30H,S+3     ;< 0 JUMP
        JC RDIN2
        CJNE A,#3AH,S+3     ;> 9 JUMP
        JC RDIN3
        CJNE A,#41H,S+3     ;< A JUMP
        JC RDIN2
        CJNE A,#47H,S+3     ;> F JUMP
        JC RDIN3
        CJNE A,#61H,S+3     ;< a JUMP
        JC RDIN2
        CJNE A,#67H,S+3     ;> f JUMP
        JNC RDIN2
RDIN3:  LCALL LCDWD
        MOV R2,#2
        LCALL DTSEC
        DJNZ R0,RDIN2

        MOV A,#0CDH         ;READ ADDRESS FROM LCD
        LCALL LCDWI
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV R1,A
        CJNE A,#80H,S+3     ;CHECK ADDRESS
        JNC RDINER
        MOV A,@R1
        MOV R1,A            ;WRITE DATA TO LCD
        MOV A,#0E1H
        LCALL LCDWI
        MOV A,R1
        LCALL HTOA
        MOV A,R2
        LCALL LCDWD
        MOV A,R3
        LCALL LCDWD
        SJMP RDIN1

RDINER: MOV DPTR,#ADDERT    ;ADDRESS ERROR
        LCALL LCDOUT
        LCALL EBEEP
        MOV R2,#20
        LCALL DTSEC
        LJMP RDIN

RDINT:  DB ' READ INTERNAL RAM '
        DB 'ADDR IN HEX:    '
        DB '(00-7F)         '
        DB 'DATA IN HEX:    '

;********** WRITE DATA IN INTERNAL RAM **********
WDIN:   MOV DPTR,#WDINT
        LCALL LCDOUT
        MOV A,#0FH
        LCALL LCDWI
WDIN1:  MOV R0,#2
        MOV A,#0CDH
        LCALL LCDWI
WDIN2:  LCALL SCAN
        CJNE A,#30H,S+3     ;< 0 JUMP
        JC WDIN2
        CJNE A,#3AH,S+3     ;> 9 JUMP
        JC WDIN3
        CJNE A,#41H,S+3     ;< A JUMP
        JC WDIN2
        CJNE A,#47H,S+3     ;> F JUMP
        JC WDIN3
        CJNE A,#61H,S+3     ;< a JUMP
        JC WDIN2
        CJNE A,#67H,S+3     ;> f JUMP
        JNC WDIN2
WDIN3:  LCALL LCDWD
        MOV R2,#2
        LCALL DTSEC
        DJNZ R0,WDIN2

        MOV A,#0CDH         ;READ ADDRESS FROM LCD
        LCALL LCDWI
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV R1,A
        CJNE A,#80H,S+3     ;CHECK ADDRESS
        JNC WDINER

WDIN4:  MOV R0,#2
```

```
    MOV A,#0E1H
    LCALL LCDWI
WDIN5:  LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC WDIN5
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC WDIN6
    CJNE A,#41H,$+3     ;< A JUMP
    JC WDIN5
    CJNE A,#47H,$+3     ;> F JUMP
    JC WDIN6
    CJNE A,#61H,$+3     ;< a JUMP
    JC WDIN5
    CJNE A,#67H,$+3     ;> f JUMP
    JNC WDIN5
WDIN6:  LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,WDIN5

    MOV A,#0E1H         ;READ DATA FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV @R1,A           ;WRITE DATA TO INTERNAL RAM
    LJMP WDIN1

WDINER: MOV DPTR,#ADDERT    ;ADDRESS ERROR
    LCALL LCDOUT
    LCALL EBEEP
    MOV R2,#20
    LCALL DTSEC
    LJMP WDIN

WDINT:  DB ' WRITE INTERNAL RAM '
    DB 'ADDR IN HEX:    '
    DB '(00-7F)         '
    DB 'DATA IN HEX:    '

;********** FIND DATA IN INTERNAL RAM **********
FNDIN:  MOV DPTR,#FNDINT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
FNDIN1: MOV R0,#2
    MOV A,#0CDH
    LCALL LCDWI
FNDIN2: LCALL SCAN

    CJNE A,#7FH,$+10    ;LEFT
    MOV R2,#2
    LCALL DTSEC
    SJMP FNDIN5
    CJNE A,#7EH,$+10    ;RIGHT
    MOV R2,#2
    LCALL DTSEC
    SJMP FNDIN4
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC FNDIN2
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC FNDIN3
    CJNE A,#41H,$+3     ;< A JUMP
    JC FNDIN2
    CJNE A,#47H,$+3     ;> F JUMP
    JC FNDIN3
    CJNE A,#61H,$+3     ;< a JUMP
    JC FNDIN2
    CJNE A,#67H,$+3     ;> f JUMP
    JNC FNDIN2
FNDIN3: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,FNDIN2

    MOV A,#0CDH         ;READ DATA FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH

    MOV FINDBUF,A
    MOV R1,#0FFH
FNDIN4: INC R1
    MOV A,@R1
    CJNE R1,#80H,$+5
    SJMP FNDIF
    CJNE A,FINDBUF,FNDIN4
    SJMP FNDIN6

FNDIN5: DEC R1
    MOV A,@R1
    CJNE R1,#0FFH,$+5
    SJMP FNDIF
    CJNE A,FINDBUF,FNDIN6

FNDIN6: MOV A,#0A1H         ;WRITE ADDRESS TO LCD
    LCALL LCDWI
    MOV A,R1
    LCALL HTOA
    MOV A,R2
    LCALL LCDWD
    MOV A,R3
    LCALL LCDWD
    LJMP FNDIN1

FNDIF:  MOV DPTR,#FNDFNT
    LCALL LCDOUT
    LCALL LBEEP
    MOV R2,#20
    LCALL DTSEC
    LJMP FNDIN

FNDINT: DB ' FIND INTERNAL RAM '
    DB 'DATA IN HEX:    '
    DB 'ADDR IN HEX:    '
    DB 00100000B
    DB 01111110B
    DB '=NEXT '
    DB 01111111B
```

```
    DB '=PREVIOUS '

;********** READ SFR **********
RDSFR:  MOV DPTR,#RDSFRT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
    MOV A,#0CAH
    LCALL LCDWI
    MOV R0,#80H
    LCALL HTON
    MOV R5,#6
RDSFR0: CLR A
    MOVC A,@A+DPTR
    LCALL LCDWD
    INC DPTR
    DJNZ R5,RDSFR0

RDSFR1: LCALL SCAN

    CJNE A,#7EH,$+10    ;RIGHT
    MOV R2,#2
    LCALL DTSEC
    SJMP RDSFR2

    CJNE A,#7FH,$+10    ;LEFT
    MOV R2,#2
    LCALL DTSEC
    SJMP RDSFR3

    CJNE A,#84H,RDSFR1  ;ENTER
    MOV R2,#2
    LCALL DTSEC
    SJMP RDSFR5

RDSFR2: INC R0
    LCALL HTON
    CJNE A,#1,RDSFR2
    SJMP RDSFR4

RDSFR3: DEC R0
    LCALL HTON
    CJNE A,#1,RDSFR3

RDSFR4: MOV A,#0CAH
    LCALL LCDWI
    MOV R5,#6
    SJMP RDSFR0

RDSFR5: MOV A,#0A0H
    LCALL LCDWI

    CJNE R0,#80H,$+8
    MOV P0,#0FFH
    MOV A,P0

    CJNE R0,#81H,$+5
    MOV A,SP

    CJNE R0,#82H,$+5
    MOV A,DPL

    CJNE R0,#83H,$+5
    MOV A,DPH

    CJNE R0,#87H,$+5
    MOV A,PCON

    CJNE R0,#88H,$+5
    MOV A,TCON

    CJNE R0,#89H,$+5
    MOV A,TMOD

    CJNE R0,#8AH,$+5
    MOV A,TL0

    CJNE R0,#8BH,$+5
    MOV A,TL1

    CJNE R0,#8CH,$+5
    MOV A,TH0

    CJNE R0,#8DH,$+5
    MOV A,TH1

    CJNE R0,#90H,$+8
    MOV P1,#0FFH
    MOV A,P1

    CJNE R0,#98H,$+5
    MOV A,SCON

    CJNE R0,#99H,$+5
    MOV A,SBUF

    CJNE R0,#0A0H,$+8
    MOV P2,#0FFH
    MOV A,P2

    CJNE R0,#0A8H,$+5
    MOV A,IE

    CJNE R0,#0B0H,$+8
    MOV P3,#0FFH
    MOV A,P3

    CJNE R0,#0B8H,$+5
    MOV A,IP

    CJNE R0,#0C8H,$+5
    MOV A,T2CON

    CJNE R0,#0CAH,$+5
    MOV A,RCAP2L

    CJNE R0,#0CBH,$+5
    MOV A,RCAP2H

    CJNE R0,#0CCH,$+5
    MOV A,TL2

    CJNE R0,#0CDH,$+5
    MOV A,TH2
```

```
    CJNE R0,#0D0H,$+5
    MOV A,PSW

    CJNE R0,#0F0H,$+5
    MOV A,B

RDSFR6: LCALL HTOA
    MOV A,R2
    LCALL LCDWD
    MOV A,R3
    LCALL LCDWD
    MOV A,#0D0H
    LCALL LCDWI
    LJMP RDSFR1

RDSFRT: DB '    READ SFR    '
    DB 'SFR NAME:       '
    DB 'DATA (HEX):     '
    DB 00100000B
    DB 01111110B
    DB '=NEXT '
    DB 01111111B
    DB '=PREVIOUS '

;********** WRITE SFR **********
WDSFR: MOV DPTR,#WDSFRT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
    MOV A,#0CAH
    LCALL LCDWI
    MOV R0,#80H
    LCALL HTON
    MOV R5,#6
WDSFR0: CLR A
    MOVC A,@A+DPTR
    LCALL LCDWD
    INC DPTR
    DJNZ R5,WDSFR0

WDSFR1: LCALL SCAN

    CJNE A,#7EH,$+10    ;RIGHT
    MOV R2,#2
    LCALL DTSEC
    SJMP WDSFR2

    CJNE A,#7FH,$+10    ;LEFT
    MOV R2,#2
    LCALL DTSEC
    SJMP WDSFR3

    CJNE A,#84H,WDSFR1    ;ENTER
    MOV R2,#2
    LCALL DTSEC
    SJMP WDSFR5

WDSFR2: INC R0
    LCALL HTON
    CJNE A,#1,WDSFR2
    SJMP WDSFR4

WDSFR3: DEC R0
    LCALL HTON
    CJNE A,#1,WDSFR3

WDSFR4: MOV A,#0CAH
    LCALL LCDWI
    MOV R5,#6
    SJMP WDSFR0

WDSFR5: MOV R1,#2
    MOV A,#0A0H
    LCALL LCDWI
WDSFR6: LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC WDSFR6
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC WDSFR7
    CJNE A,#41H,$+3     ;< A JUMP
    JC WDSFR6
    CJNE A,#47H,$+3     ;> F JUMP
    JC WDSFR7
    CJNE A,#61H,$+3     ;< a JUMP
    JC WDSFR6
    CJNE A,#67H,$+3     ;> f JUMP
    JNC WDSFR6

WDSFR7: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R1,WDSFR6

    MOV A,#0A0H
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH

    CJNE R0,#80H,$+5
    MOV P0,A

    CJNE R0,#81H,$+5
    MOV SP,A

    CJNE R0,#82H,$+5
    MOV DPL,A

    CJNE R0,#83H,$+5
    MOV DPH,A

    CJNE R0,#87H,$+5
    MOV PCON,A

    CJNE R0,#88H,$+5
    MOV TCON,A

    CJNE R0,#89H,$+5
    MOV TMOD,A

    CJNE R0,#8AH,$+5
```

```
    MOV TL0,A

    CJNE R0,#8BH,S+5
    MOV TL1,A

    CJNE R0,#8CH,S+5
    MOV TH0,A

    CJNE R0,#8DH,S+5
    MOV TH1,A

    CJNE R0,#90H,S+5
    MOV P1,A

    CJNE R0,#98H,S+5
    MOV SCON,A

    CJNE R0,#99H,S+5
    MOV SBUF,A

    CJNE R0,#0A0H,S+5
    MOV P2,A

    CJNE R0,#0A8H,S+5
    MOV IE,A

    CJNE R0,#0B0H,S+5
    MOV P3,A

    CJNE R0,#0B8H,S+5
    MOV IP,A

    CJNE R0,#0C8H,S+5
    MOV T2CON,A

    CJNE R0,#0CAH,S+5
    MOV RCAP2L,A

    CJNE R0,#0CBH,S+5
    MOV RCAP2H,A

    CJNE R0,#0CCH,S+5
    MOV TL2,A

    CJNE R0,#0CDH,S+5
    MOV TH2,A

    CJNE R0,#0D0H,S+5
    MOV PSW,A

    CJNE R0,#0F0H,S+5
    MOV B,A

WDSFR8: MOV A,#0D0H
    LCALL LCDWI
    LJMP WDSFR1

WDSFRT: DB '   WRITE SFR   '
    DB 'SFR NAME:     '
    DB 'DATA (HEX):   '
    DB 00000000B
    DB 01111110B
    DB '=NEXT '
    DB 01111111B
    DB '=PREVIOUS '

;********** LOAD PROGRAM **********
LOAD:  MOV DPTR,#8000H
    CLR A
LOADC:  MOVX @DPTR,A
    INC DPTR
    MOV R2,DPH
    CJNE R2,#0A0H,LOADC

    MOV DPTR,#LOADT1
    LCALL LCDOUT
    CLR RI
    MOV MEMADD,#0
    MOV MEMADD+1,#0
    CLR STMF
LOAD0:  JB INT1,LOAD01
    LCALL HBEEP
    LJMP MAIN
LOAD01: JNB RI,LOAD0
    LCALL RBYTE
    CJNE A,#0DH,S+5     ;DON'T CARE 0DH (CR)
    SJMP LOAD0
    CJNE A,#0AH,S+5     ;DON'T CARE 0AH (LF)
    SJMP LOAD0
    CJNE A,#'O',S+6
    LJMP LOADO
    CJNE A,#'S',S+6
    LJMP LOADS
    CJNE A,#':',S+6
    LJMP LOAD11
    LJMP LOADE

LOAD1:  LCALL RBYTE
    CJNE A,#':',LOAD1
LOAD11: MOV R7,#0           ;CLEAR CHECKSUM
    LCALL RBYTEHC       ;BYTE COUNT
    MOV R5,A            ;COUNTER
    LCALL RBYTEHC
    MOV DPH,A           ;ADDRESS HIGH
    LCALL RBYTEHC
    MOV DPL,A           ;ADDRESS LOW
    LCALL RBYTEHC
    CJNE A,#00H,LOAD3   ;END OF FILE

    JNB STMF,LOAD12     ;CHECK FIRST STREAM-OFFSET ADDRESS
    MOV R2,DPH
    MOV R3,DPL
    MOV DPH,MEMADD
    MOV DPL,MEMADD+1
    LCALL DPSUB
    MOV MEMADD,DPH
    MOV MEMADD+1,DPL
    MOV DPH,R2
    MOV DPL,R3
    CLR STMF
LOAD12: MOV R2,MEMADD       ;OFFSET PROCESS
    MOV R3,MEMADD+1
    LCALL DPADD
```

```
LOAD2: LCALL RBYTEHC      ;DATA LOOP
    MOVX @DPTR,A
    INC DPTR
    DJNZ R5,LOAD2
    MOV A,R7
    CPL A
    INC A
    MOV B,A
    LCALL RBYTEHC
    CJNE A,B,LOADE
    LCALL RBYTE       ;CR
    LCALL RBYTE       ;LF
    LJMP LOAD1

LOAD3: MOV A,R7
    CPL A
    INC A
    MOV B,A
    LCALL RBYTEH
    CJNE A,B,LOADE
    LCALL RBYTE       ;CR
    LCALL RBYTE       ;LF
    MOV DPTR,#LOADT3
    LCALL LCDOUT3
    MOV R2,#20
    LCALL DTSEC
    LCALL HBEEP
    LJMP MAIN

LOADE: MOV DPTR,#LOADT2
    LCALL LCDOUT3
    LCALL EBEEP
    MOV R2,#20
    LCALL DTSEC
    LJMP MAIN

LOADS: SETB STMF          ;STREAM RECEIVE
LOAD0: LCALL RBYTEH       ;OFFSET RECEIVE
    MOV MEMADD,A      ;ADDRESS HIGH
    LCALL RBYTEH
    MOV MEMADD+1,A    ;ADDRESS LOW
    LCALL RBYTE       ;CR
    LCALL RBYTE       ;LF
    LJMP LOAD0

LOADT1: DB ' LOAD HEX FILE '
    DB '               '
    DB ' LOADING... '
    DB '               '
LOADT2: DB ' DOWNLOAD ERROR '
LOADT3: DB ' FINISH LOAD '

;********** UPLOAD PROGRAM **********
UPLD: MOV DPTR,#UPLDT1
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
UPLD1: MOV R0,#4
    MOV A,#0D0H
    LCALL LCDWI
UPLD2: LCALL SCAN
    CJNE A,#30H,S+3   ;< 0 JUMP
    JC UPLD2
    CJNE A,#3AH,S+3   ;> 9 JUMP
    JC UPLD3
    CJNE A,#41H,S+3   ;< A JUMP
    JC UPLD2
    CJNE A,#47H,S+3   ;> F JUMP
    JC UPLD3
    CJNE A,#61H,S+3   ;< a JUMP
    JC UPLD2
    CJNE A,#67H,S+3   ;> f JUMP
    JNC UPLD2
UPLD3: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,UPLD2

    MOV A,#0D0H       ;READ START ADDRESS FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV STTADD,A
    CJNE A,#80H,S+3   ;CHECK ADDRESS
    JNC S+5
    LJMP UPLDER
    CJNE A,#0A0H,S+3
    JC S+5
    LJMP UPLDER
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV STTADD+1,A

UPLD4: MOV R0,#4
    MOV A,#0A4H
    LCALL LCDWI
UPLD5: LCALL SCAN
    CJNE A,#30H,S+3   ;< 0 JUMP
    JC UPLD5
    CJNE A,#3AH,S+3   ;> 9 JUMP
    JC UPLD6
    CJNE A,#41H,S+3   ;< A JUMP
    JC UPLD5
    CJNE A,#47H,S+3   ;> F JUMP
    JC UPLD6
    CJNE A,#61H,S+3   ;< a JUMP
    JC UPLD5
    CJNE A,#67H,S+3   ;> f JUMP
    JNC UPLD5
UPLD6: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,UPLD5

    MOV A,#0A4H       ;READ END ADDRESS FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
```

```
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV ENDADD,A
    CJNE A,#80H,$+3     ;CHECK ADDRESS
    JNC $+5
    LJMP UPLDER
    CJNE A,#0A0H,$+3
    JC $+5
    LJMP UPLDER
    MOV DPH,STTADD
    CJNE A,DPH,$+3
    JNC $+5
    LJMP UPLDER
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV ENDADD+1,A
    MOV DPL,STTADD+1
    CJNE A,DPL,$+3
    JNC $+5
    LJMP UPLDER

UPLD7:  MOV DPTR,#UPLDT2
    LCALL LCDOUT
    MOV DPH,STTADD
    MOV DPL,STTADD+1
    CLR SENDEN          ;END FLAG

UPLD8:  MOV R7,#0           ;START CHECKSUM
    MOV A,#':'          ;FIRST BYTE
    LCALL SBYTE
    LCALL UPLDC         ;CHECK LOOP COUNTER
    MOV A,R6
    LCALL SBYTEHC
    MOV A,DPH
    LCALL SBYTEHC       ;ADDRESS
    MOV A,DPL
    LCALL SBYTEHC
    CLR A
    LCALL SBYTEHC

UPLD9:  MOVX A,@DPTR
    LCALL SBYTEHC       ;DATA
    INC DPTR
    DJNZ R6,UPLD9
    MOV A,R7            ;CHECKSUM
    CPL A
    INC A
    LCALL SBYTEH
    LCALL SLF
    JNB SENDEN,UPLD8

    MOV DPTR,#UPLDT4
    LCALL SBLOCK
    MOV A,#1AH          ;^Z END-OF-FILE
    LCALL SBYTE
    LCALL HBEEP
    MOV DPTR,#UPLDT3
    LCALL LCDOUT3
    MOV R2,#20
    LCALL DTSEC
    LJMP MAIN

UPLDC:  MOV A,DPH       ;CHECK LINE LOOP
    XCH A,R2
    MOV DPH,A
    MOV A,DPL
    XCH A,R3
    MOV DPL,A
    MOV DPH,ENDADD
    MOV DPL,ENDADD+1
    LCALL DPSUB
    MOV A,DPH
    JNZ UPLDC3
    MOV A,DPL
    CJNE A,#0FH,$+6
    LJMP UPLDC2
    JNC UPLDC3
    SETB SENDEN         ;LAST LINE (<10H LOOP)
    MOV R6,DPL
    INC R6
    MOV DPH,R2
    MOV DPL,R3
    RET

UPLDC2: SETB SENDEN         ;LAST LINE (10H LOOP)
UPLDC3: MOV DPH,R2          ;10H LOOP
    MOV DPL,R3
    MOV R6,#10H         ;LOOP COUNTER
    RET

UPLDER: MOV DPTR,#ADDERT    ;ADDRESS ERROR
    LCALL LCDOUT
    MOV R2,#20
    LCALL DTSEC
    LJMP UPLD

UPLDT1: DB ' UPLOAD HEX FILE  '
    DB 'START ADDR(HEX):  '
    DB 'END   ADDR(HEX):  '
    DB '(8000-9FFF)       '
UPLDT2: DB ' UPLOAD HEX FILE  '
    DB '                  '
    DB ' UPLOADING...     '
    DB '                  '
UPLDT3: DB ' FINISH UPLOAD    '
UPLDT4: DB ':00000001FF',0DH

;********** UPLOAD ASSEMBLY FILE **********
UPAS:  MOV DPTR,#UPAST1
    LCALL LCDOUT
    MOV DPTR,#0FBFFH
UPAS1:  MOVX A,@DPTR
    CJNE A,#20H,$+8
    LCALL DPDEC
    SJMP UPAS1
    MOV ENDADD,DPH
    MOV ENDADD+1,DPL

    MOV DPTR,#UPAST2
    MOV R7,#9
```

```
UPAS2:  CLR A
        MOVC A,@A+DPTR
        LCALL SBYTE
        INC DPTR
        DJNZ R7,UPAS2
        LCALL SLF

        MOV DPTR,#0A004H
        CLR SENDEN
UPAS3:  MOV R7,#20
UPAS4:  MOVX A,@DPTR
        LCALL SBYTE
        INC DPTR
        DJNZ R7,UPAS4
        LCALL UPASC3
        CJNE A,#1,UPAS6

        MOV A,#3BH          ;HAVE COMMENT
        LCALL SBYTE
        MOV R7,#16
UPAS5:  MOVX A,@DPTR
        LCALL SBYTE
        INC DPTR
        DJNZ R7,UPAS5
        INC DPTR
        INC DPTR
        INC DPTR
        INC DPTR
        SJMP UPAS7
UPAS6:  MOV R2,#0           ;NO HAVE COMMENT
        MOV R3,#20
        LCALL DPADD
UPAS7:  LCALL SLF
        LCALL UPASC1
        JNB SENDEN,UPAS3

        MOV DPTR,#UPAST3
        MOV R7,#3
UPAS8:  CLR A
        MOVC A,@A+DPTR
        LCALL SBYTE
        INC DPTR
        DJNZ R7,UPAS8
        LCALL SLF

        MOV A,#1AH          ;^Z END-OF-FILE
        LCALL SBYTE
        LCALL HBEEP
        MOV DPTR,#UPLDT3
        LCALL LCDOUT3
        MOV R2,#20
        LCALL DTSEC
        LJMP MAIN

UPASC1: MOV R2,DPH          ;CHECK END
        MOV R3,DPL
        MOV DPH,ENDADD
        MOV DPL,ENDADD+1
        MOV A,R2
        CJNE A,DPH,S+3
        JC UPASC2
        CJNE A,DPH,UPASC11
        MOV A,R3
        CJNE A,DPL,S+3
        JC UPASC2
UPASC11: SETB SENDEN
UPASC2:  MOV DPH,R2
         MOV DPL,R3
         RET

UPASC3: MOV R6,#16          ;CHECK HAVE COMMENT
        MOV R2,DPH
        MOV R3,DPL
UPASC4: MOVX A,@DPTR
        CJNE A,#20H,S+9
        INC DPTR
        DJNZ R6,UPASC4
        CLR A
        SJMP S+4
        MOV A,#1
        MOV DPH,R2
        MOV DPL,R3
        RET

UPAST1: DB 'UPLOAD ASSEMBLY FILE'
        DB '                    '
        DB '   UPLOADING...     '
        DB '                    '
UPAST2: DB 'ORG 8000H'
UPAST3: DB 'END'

;********** SET BAUD RATE **********
BAUD:   MOV B,PCON
        MOV R0,B
        MOV DPTR,#BAUDT
        LCALL LCDOUT
BAUD1:  LCALL SCAN
        MOV B,R0

        CJNE A,#31H,S+11    ;1200
        LCALL BLACK1
        CLR SMOD
        MOV TH1,#BRAT12

        CJNE A,#32H,S+11    ;2400
        LCALL BLACK2
        CLR SMOD
        MOV TH1,#BRAT24

        CJNE A,#33H,S+11    ;4800
        LCALL BLACK3
        CLR SMOD
        MOV TH1,#BRAT48

        CJNE A,#34H,S+11    ;9600
        LCALL BLACK4
        CLR SMOD
        MOV TH1,#BRAT96

        CJNE A,#35H,S+11    ;19200
        LCALL BLACK5
        SETB SMOD
        MOV TH1,#BRAT96
```

```
    MOV PCON,B
    LJMP MAIN

BAUDT:  DB '  SET BAUD RATE   '
    DB '1.1200   4.9600   '
    DB '2.2400   5.19200  '
    DB '3.4800            '

;********** INPUT PORT **********
INPUT:  MOV DPTR,#INPUTT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
INPUT1: MOV A,#0CEH
    LCALL LCDWI
    LCALL SCAN          ;READ PORT FROM LCD
    CJNE A,#61H,$+5     ;a JUMP
    SJMP INPUT2
    CJNE A,#63H,$+5     ;c JUMP
    SJMP INPUT2
    CJNE A,#41H,$+5     ;A JUMP
    SJMP INPUT2
    CJNE A,#43H,$+5     ;C JUMP
    SJMP INPUT2
    SJMP INPUT1

INPUT2: MOV R1,A
    LCALL LCDWD
    CJNE R1,#61H,$+6    ;READ DATA FROM PORT
    MOV DPTR,#U8255A
    CJNE R1,#63H,$+6
    MOV DPTR,#U8255C
    CJNE R1,#41H,$+6
    MOV DPTR,#U8255A
    MOV DPTR,#U8255C
    MOVX A,@DPTR
    MOV R1,A            ;WRITE DATA TO LCD
    MOV A,#0A2H
    LCALL LCDWI
    MOV A,R1
    LCALL HTOA
    MOV A,R2
    LCALL LCDWD
    MOV A,R3
    LCALL LCDWD
    SJMP INPUT1

INPUTT: DB '   INPUT PORT    '
    DB 'PORT(B,C lo):    '
    DB 'DATA  (HEX):     '
    DB '                 '

;********** OUTPUT PORT **********
OUTPUT:  MOV DPTR,#OUTPUTT
    LCALL LCDOUT
    MOV A,#0FH
    LCALL LCDWI
OUTPUT1: MOV A,#0CEH
    LCALL LCDWI
    LCALL SCAN          ;READ PORT FROM LCD
    CJNE A,#61H,$+5     ;a JUMP
    SJMP OUTPUT2
    CJNE A,#63H,$+5     ;c JUMP
    SJMP OUTPUT2
    CJNE A,#41H,$+5     ;A JUMP
    SJMP OUTPUT2
    CJNE A,#43H,$+5     ;C JUMP
    SJMP OUTPUT2
    SJMP OUTPUT1
OUTPUT2: MOV R1,A
    LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
OUTPUT3: MOV R0,#2
    MOV A,#0A2H
    LCALL LCDWI
OUTPUT4: LCALL SCAN
    CJNE A,#30H,$+3     ;< 0 JUMP
    JC OUTPUT4
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC OUTPUT5
    CJNE A,#41H,$+3     ;< A JUMP
    JC OUTPUT4
    CJNE A,#47H,$+3     ;> F JUMP
    JC OUTPUT5
    CJNE A,#61H,$+3     ;< a JUMP
    JC OUTPUT4
    CJNE A,#67H,$+3     ;> f JUMP
    JNC OUTPUT4
OUTPUT5: LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,OUTPUT4

    MOV A,#0A2H         ;READ DATA FROM LCD
    LCALL LCDWI
    LCALL LCDRD
    MOV R2,A
    LCALL LCDRD
    MOV R3,A
    LCALL ATOH
    MOV R5,A
    CJNE R1,#61H,$+6    ;WRITE DATA TO PORT
    MOV DPTR,#U8255A
    CJNE R1,#63H,$+6
    MOV DPTR,#U8255C
    CJNE R1,#41H,$+6
    MOV DPTR,#U8255A
    CJNE R1,#43H,$+6
    MOV DPTR,#U8255C
    MOV A,R5
    MOVX @DPTR,A
    LJMP OUTPUT1

OUTPUTT: DB '   OUTPUT PORT   '
    DB 'PORT(A,C up):    '
    DB 'DATA  (HEX):     '
    DB '                 '

;********** CHECK CIRCUIT **********
CHKCC:  MOV DPTR,#CHKCCT
    LCALL LCDOUT
CHKCC1: LCALL SCAN
    MOV DPTR,#S8255C
```

```
        MOVX A,@DPTR
        CLR ACC.6
        MOVX @DPTR,A
        MOVX A,@DPTR
        JNB ACC.3,$+11
        MOV DPTR,#CHKCCNS
        LCALL LCDOUT3
        SJMP CHKCC1
        MOV DPTR,#S8255C
        MOVX A,@DPTR
        SETB ACC.6
        MOVX @DPTR,A
        MOVX A,@DPTR
        JB ACC.3,$+11
        MOV DPTR,#CHKCCNS
        LCALL LCDOUT3
        SJMP CHKCC1
        MOV DPTR,#CHKCCS
        LCALL LCDOUT3
        LCALL HBEEP
        SJMP CHKCC1

CHKCCT: DB '  CHECK CIRCUIT  '
        DB '                '
        DB '                '
        DB '                '
CHKCCNS: DB '    NO SHORT    '
CHKCCS: DB '     SHORT      '

;********** PLAY SONG **********
PSONG:  MOV DPTR,#PSONGT1
        LCALL LCDOUT
PSONG2: LCALL SCAN

        CJNE A,#32H,$+15      ;SPIDER
        LCALL BLACK2
        MOV DPTR,#PSONGT2
        LCALL SONG
        LJMP MAIN

        CJNE A,#31H,PSONG2    ;HAPPY BIRTHDAY
        LCALL BLACK1
        MOV DPTR,#PSONGT6
        LCALL LCDOUT
        MOV DPTR,#PSONGT3
        LCALL SONG
        MOV DPTR,#PSONGT7
        LCALL LCDOUT
        MOV DPTR,#PSONGT4
        LCALL SONG
        MOV DPTR,#PSONGT6
        LCALL LCDOUT
        MOV DPTR,#PSONGT5
        LCALL SONG
        LJMP MAIN

PSONGT1: DB '  LISTEN TO SONG  '
        DB '1.HAPPY BIRTH DAY  '
        DB '2.SPIDER           '
        DB '                   '
PSONGT2: DB 0CH,04H,0CH,04H
        DB 0CH,04H,0EH,04H
        DB 10H,04H,25H,01H
        DB 10H,04H,10H,04H
        DB 0EH,04H,0CH,04H
        DB 0EH,04H,10H,04H
        DB 0CH,04H,25H,01H
        DB 10H,04H,10H,04H
        DB 10H,04H,11H,04H
        DB 13H,04H,25H,01H
        DB 13H,04H,13H,04H
        DB 11H,04H,10H,04H
        DB 11H,04H,13H,04H
        DB 10H,04H,25H,01H
        DB 0CH,04H,0CH,04H
        DB 0CH,04H,0EH,04H
        DB 10H,04H,10H,04H,25H,01H
        DB 10H,04H,0EH,04H
        DB 0CH,04H,0EH,04H
        DB 10H,04H,0CH,04H
        DB 0FFH

PSONGT3: DB 05H,04H,05H,04H
        DB 07H,04H,05H,04H
        DB 0AH,04H,09H,04H,25H,01H
        DB 05H,04H,05H,04H
        DB 07H,04H,05H,04H
        DB 0CH,04H,0AH,04H,25H,01H,0FFH
PSONGT4: DB 05H,04H,05H,04H
        DB 13H,04H,11H,04H
        DB 0CH,04H,0AH,04H
        DB 09H,04H,07H,04H,25H,01H,0FFH
PSONGT5: DB 11H,04H,11H,04H
        DB 0EH,04H,0AH,04H
        DB 0CH,04H,0AH,04H
        DB 0FFH
PSONGT6: DB '                '
        DB '  HAPPY BIRTHDAY  '
        DB '     TO YOU     '
        DB '                '
PSONGT7: DB '                '
        DB '  HAPPY BIRTHDAY  '
        DB '  HAPPY BIRTHDAY  '
        DB '                '

;********** QUIT **********
QUIT:   MOV DPTR,#QUITT1
        LCALL LCDOUT
        MOV A,#0FH
        LCALL LCDWI
QUIT1:  MOV R0,#4
        MOV A,#9CH
        LCALL LCDWI
        LCALL SCAN
        CJNE A,#84H,$+13
        CLR POWF
        MOV DPTR,#QUITT2
        LCALL LCDOUT
        SJMP $

        PUSH ACC
        MOV DPTR,#QUITT3
        LCALL LCDOUT4
```

```
    MOV A,#9CH
    LCALL LCDWI
    POP ACC
    SJMP QUIT3
QUIT2:  LCALL SCAN
QUIT3:  CJNE A,#30H,$+3     ;< 0 JUMP
    JC QUIT2
    CJNE A,#3AH,$+3     ;> 9 JUMP
    JC QUIT4
    CJNE A,#41H,$+3     ;< A JUMP
    JC QUIT2
    CJNE A,#5BH,$+3     ;> Z JUMP
    JC QUIT4
    CJNE A,#61H,$+3     ;< a JUMP
    JC QUIT2
    CJNE A,#7BH,$+3     ;> z JUMP
    JNC QUIT2
QUIT4:  LCALL LCDWD
    MOV R2,#2
    LCALL DTSEC
    DJNZ R0,QUIT2

    MOV A,#9CH
    LCALL LCDWI
    LCALL LCDRD
    MOV POWMEM,A
    LCALL LCDRD
    MOV POWMEM+1,A
    LCALL LCDRD
    MOV POWMEM+2,A
    LCALL LCDRD
    MOV POWMEM+3,A
    MOV DPTR,#QUITT4
    LCALL LCDOUT4

    LCALL SCAN
    CJNE A,#84H,$-3
    SETB POWF
    MOV DPTR,#QUITT5
    LCALL LCDOUT
    SJMP $

QUITT1: DB '   QUIT PROGRAM   '
    DB '  PRESS PASSWORD  '
    DB '                  '
    DB '   ENTER=NONE     '
QUITT2: DB '                  '
    DB '   QUIT PROGRAM   '
    DB '   NO PASSWORD    '
    DB '                  '
QUITT3: DB '                  '
QUITT4: DB '   ENTER=QUIT     '
QUITT5: DB '                  '
    DB '   QUIT PROGRAM   '
    DB '   WITH PASSWORD  '
    DB '                  '

;********** LCD SET SUB **********
LCDSET: MOV A,#00111000B    ;FUNCTION SET
    LCALL LCDWI
    MOV A,#00001111B    ;DISPLAY ON/OFF
    LCALL LCDWI
    MOV A,#00000110B    ;ENTRY MODE SET
    LCALL LCDWI
    MOV A,#00000001B    ;CLEAR
    LCALL LCDWI
    RET
;********** LCD WRITE INSTRUCTION SUB **********
LCDWI:  PUSH DPH
    PUSH DPL
    MOV DPTR,#LCDWRC
    MOVX @DPTR,A
    MOV DPTR,#LCDRDC
LCDWI1: MOVX A,@DPTR        ;WAIT FOR BF=0
    JB ACC.7,LCDWI1
    POP DPL
    POP DPH
    RET
;********** LCD READ DATA SUB **********
LCDRD:  PUSH DPH
    PUSH DPL
    MOV DPTR,#LCDRDD
    MOVX A,@DPTR
    MOV B,A
    MOV DPTR,#LCDRDC
LCDRD1: MOVX A,@DPTR        ;WAIT FOR BF=0
    JB ACC.7,LCDRD1
    MOV A,B
    POP DPL
    POP DPH
    RET
;********** LCD WRITE DATA SUB **********
LCDWD:  PUSH DPH
    PUSH DPL
    MOV DPTR,#LCDWRD
    MOVX @DPTR,A
    MOV DPTR,#LCDRDC
LCDWD1: MOVX A,@DPTR        ;WAIT FOR BF=0
    JB ACC.7,LCDWD1
    POP DPL
    POP DPH
    RET
;********** LCD OUT SUB **********
LCDOUT:  MOV A,#80H     ;4 LINES
    LCALL LCDOUTS
    MOV A,#0C0H
    LCALL LCDOUTS
    MOV A,#94H
    LCALL LCDOUTS
    MOV A,#0D4H
    LCALL LCDOUTS
    MOV A,#0CH
    LCALL LCDWI
    RET
LCDOUT1:  MOV A,#80H        ;LINE 1
    LCALL LCDOUTS
    RET
LCDOUT2:  MOV A,#0C0H       ;LINE 2
    LCALL LCDOUTS
    RET
LCDOUT3:  MOV A,#94H        ;LINE 3
    LCALL LCDOUTS
    RET
LCDOUT4:  MOV A,#0D4H       ;LINE 4
```

```
        LCALL LCDOUTS
        RET
LCDOUTS:  LCALL LCDWI       ;LOAD ONE LINE
        MOV R2,#20
LCDOUTS1: CLR A
        MOVC A,@A+DPTR
        LCALL LCDWD
        INC DPTR
        DJNZ R2,LCDOUTS1
        RET

;********** DISPLAY CHOICE SUB **********
BLACK:  MOV A,#0FH
        LCALL LCDWI
        RET
BLACK1: LCALL BLACK
        MOV A,#0C0H
        SJMP BLACKS
BLACK2: LCALL BLACK
        MOV A,#94H
        SJMP BLACKS
BLACK3: LCALL BLACK
        MOV A,#0D4H
        SJMP BLACKS
BLACK4: LCALL BLACK
        MOV A,#0CAH
        SJMP BLACKS
BLACK5: LCALL BLACK
        MOV A,#9EH
        SJMP BLACKS
BLACK6: LCALL BLACK
        MOV A,#0DEH
BLACKS: LCALL LCDWI
        MOV R2,#10
        LCALL DTSEC
        RET

;********** SCAN KEYBOARD SUB **********
SCAN:   MOV SCAADD,DPH
        MOV SCAADD+1,DPL
        CLR A
SCAN11:  MOV DPTR,#KBD
        MOVX @DPTR,A
        MOV R2,A
        MOV DPTR,#S8255B
        MOVX A,@DPTR
        CPL A
        JZ SCAN13
        MOV R3,A
        LCALL SCANC
        CJNE R2,#0,SCAN111
        CJNE R3,#10H,$+6
        CLR A
        SJMP SCAN11
        JNB SHIFTF,$+8
        CJNE R3,#08H,SCAN12
        SJMP SCAN13
        JNB CTRLF,$+8
        CJNE R3,#04H,SCAN12
        SJMP SCAN13
SCAN111: CJNE R2,#08H,SCAN112
        CJNE A,#08H,SCAN12
        CLR A
        SJMP SCAN11
SCAN112: CJNE R2,#09H,SCAN12
        CJNE A,#08H,SCAN12
        CLR A
        SJMP SCAN11
SCAN12:  MOV DPH,SCAADD
        MOV DPL,SCAADD+1
        MOV B,A
        JB SHIFTF,$+6
        LCALL FBEEP
        MOV A,B
        RET
SCAN13:  MOV A,R2
        INC A
        CJNE A,#8,SCAN11
        JB SCANOL,SCAN12
        SJMP SCAN
;********** SCAN KEY ONE LOOP SUB **********
SCANONE: SETB SCANOL
        MOV R2,#1
        LCALL DTSEC
        LCALL SCAN
        CLR SCANOL
        RET
;********** SCAN KEY REMEMBER CHARACTER **********
SCANC:  JNB SHIFTF,$+6
        LJMP SCANC2
        JNB CTRLF,$+6
        LJMP SCANC4
        JNB CAPF,$+6
        LJMP SCANC3
SCANC10: CJNE R2,#0,SCANC11
        CJNE R3,#20H,$+10    ;MON
        JNB POWF,$+4
        RET
        LJMP MAIN

        CJNE R3,#10H,$+14    ;CAP LOCK
        CPL CAPF
        MOV R2,#2
        LCALL DTSEC
        MOV R3,#10H
        MOV R2,#0

        CJNE R3,#08H,$+13    ;SHIFT
        SETB SHIFTF
        LCALL SCANONE
        CLR SHIFTF
        MOV R2,#08H
        RET

        CJNE R3,#04H,$+13    ;SHIFT
        SETB SHIFTF
        LCALL SCANONE
        CLR SHIFTF
        MOV R2,#08H
        RET

        CJNE R3,#02H,$+5     ;8
        MOV A,#38H
```

```
        CJNE R3,#01H,S+5    ;K
        MOV A,#6BH

SCANC11: CJNE R2,#1,SCANC12
        CJNE R3,#20H,S+5    ;I
        MOV A,#31H

        CJNE R3,#10H,S+5    ;Q
        MOV A,#71H

        CJNE R3,#08H,S+5    ;A
        MOV A,#61H

        CJNE R3,#04H,S+5    ;Z
        MOV A,#7AH

        CJNE R3,#02H,S+5    ;9
        MOV A,#39H

        CJNE R3,#01H,S+5    ;L
        MOV A,#6CH

SCANC12: CJNE R2,#2,SCANC13
        CJNE R3,#20H,S+5    ;2
        MOV A,#32H

        CJNE R3,#10H,S+5    ;W
        MOV A,#77H

        CJNE R3,#08H,S+5    ;S
        MOV A,#73H

        CJNE R3,#04H,S+5    ;X
        MOV A,#78H

        CJNE R3,#02H,S+5    ;0
        MOV A,#30H

        CJNE R3,#01H,S+5    ;UP
        MOV A,#80H

SCANC13: CJNE R2,#3,SCANC14
        CJNE R3,#20H,S+5    ;3
        MOV A,#33H

        CJNE R3,#10H,S+5    ;E
        MOV A,#65H

        CJNE R3,#08H,S+5    ;D
        MOV A,#64H

        CJNE R3,#04H,S+5    ;C
        MOV A,#63H

        CJNE R3,#02H,S+5    ;BACK
        MOV A,#82H

        CJNE R3,#01H,S+5    ;ENTER
        MOV A,#84H

SCANC14: CJNE R2,#4,SCANC15
        CJNE R3,#20H,S+5    ;4
        MOV A,#34H

        CJNE R3,#10H,S+5    ;R
        MOV A,#72H

        CJNE R3,#08H,S+5    ;F
        MOV A,#66H

        CJNE R3,#04H,S+5    ;V
        MOV A,#76H

        CJNE R3,#02H,S+5    ;1
        MOV A,#69H

        CJNE R3,#01H,S+5    ;SPACE
        MOV A,#20H

SCANC15: CJNE R2,#5,SCANC16
        CJNE R3,#20H,S+5    ;5
        MOV A,#35H

        CJNE R3,#10H,S+5    ;T
        MOV A,#74H

        CJNE R3,#08H,S+5    ;G
        MOV A,#67H

        CJNE R3,#04H,S+5    ;B
        MOV A,#62H

        CJNE R3,#02H,S+5    ;O
        MOV A,#6FH

        CJNE R3,#01H,S+5    ;LEFT
        MOV A,#7FH

SCANC16: CJNE R2,#6,SCANC17
        CJNE R3,#20H,S+5    ;6
        MOV A,#36H

        CJNE R3,#10H,S+5    ;Y
        MOV A,#79H

        CJNE R3,#08H,S+5    ;H
        MOV A,#68H

        CJNE R3,#04H,S+5    ;N
        MOV A,#6EH

        CJNE R3,#02H,S+5    ;P
        MOV A,#70H

        CJNE R3,#01H,S+5    ;DOWN
        MOV A,#81H

SCANC17: CJNE R2,#7,SCANC18
        CJNE R3,#20H,S+5    ;7
        MOV A,#37H

        CJNE R3,#10H,S+5    ;U
        MOV A,#75H
```

```
        CJNE R3,#08H,S+5    ;J
        MOV A,#6AH

        CJNE R3,#04H,S+5    ;M
        MOV A,#6DH

        CJNE R3,#02H,S+5    ;DEL
        MOV A,#83H

        CJNE R3,#01H,S+5    ;RIGHT
        MOV A,#7EH
SCANC18: RET

SCANC2:  CJNE R2,#0,SCANC21
        CJNE R3,#28H,S+5
        MOV A,#08H

        CJNE R3,#18H,S+5
        MOV A,#08H

        CJNE R3,#0CH,S+5
        MOV A,#08H

        CJNE R3,#0AH,S+5    ;*
        MOV A,#'*'

        CJNE R3,#09H,S+5    ;_
        MOV A,#'_'

SCANC21: CJNE R2,#1,SCANC22
        CJNE R3,#20H,S+5    ;!
        MOV A,#21H

        CJNE R3,#10H,S+5    ;{
        MOV A,#7BH

        CJNE R3,#08H,S+5
        MOV A,#08H

        CJNE R3,#04H,S+5
        MOV A,#08H

        CJNE R3,#02H,S+5    ;(
        MOV A,#'('

        CJNE R3,#01H,S+5    ;|
        MOV A,#'|'

SCANC22: CJNE R2,#2,SCANC23
        CJNE R3,#20H,S+5    ;@
        MOV A,#40H

        CJNE R3,#10H,S+5    ;}
        MOV A,#7DH

        CJNE R3,#08H,S+5
        MOV A,#08H

        CJNE R3,#04H,S+5
        MOV A,#08H

        CJNE R3,#02H,S+5    ;)
        MOV A,#')'

        CJNE R3,#01H,S+5
        MOV A,#08H

SCANC23: CJNE R2,#3,SCANC24
        CJNE R3,#20H,S+5    ;#
        MOV A,#23H

        CJNE R3,#10H,S+5    ;[
        MOV A,#5BH

        CJNE R3,#08H,S+5
        MOV A,#08H

        CJNE R3,#04H,S+5
        MOV A,#08H

        CJNE R3,#02H,S+5
        MOV A,#08H

        CJNE R3,#01H,S+5
        MOV A,#08H

SCANC24: CJNE R2,#4,SCANC25
        CJNE R3,#20H,S+5    ;$
        MOV A,#24H

        CJNE R3,#10H,S+5    ;]
        MOV A,#5DH

        CJNE R3,#08H,S+5    ;'
        MOV A,#27H

        CJNE R3,#04H,S+5    ;;
        MOV A,#3BH

        CJNE R3,#02H,S+5    ;=
        MOV A,#3DH

        CJNE R3,#01H,S+5
        MOV A,#08H

SCANC25: CJNE R2,#5,SCANC26
        CJNE R3,#20H,S+5    ;%
        MOV A,#25H

        CJNE R3,#10H,S+5    ;<
        MOV A,#3CH

        CJNE R3,#08H,S+5    ;"
        MOV A,#22H

        CJNE R3,#04H,S+5    ;:
        MOV A,#3AH

        CJNE R3,#02H,S+5    ;/
        MOV A,#2FH

        CJNE R3,#01H,S+5
        MOV A,#08H
```

```
SCANC26: CJNE R2,#6,SCANC27
    CJNE R3,#20H,S+5    ;^
    MOV A,#5EH

    CJNE R3,#10H,S+5    ;>
    MOV A,#3EH

    CJNE R3,#08H,S+5    ;?
    MOV A,#3FH

    CJNE R3,#04H,S+5    ;:
    MOV A,#2CH

    CJNE R3,#02H,S+5    ;\
    MOV A,#'\'

    CJNE R3,#01H,S+5
    MOV A,#08H

SCANC27: CJNE R2,#7,SCANC28
    CJNE R3,#20H,S+5    ;&
    MOV A,#26H

    CJNE R3,#10H,S+5    ;*
    MOV A,#2BH

    CJNE R3,#08H,S+5    ;-
    MOV A,#2DH

    CJNE R3,#04H,S+5    ;=
    MOV A,#'='

    CJNE R3,#02H,S+5
    MOV A,#08H

    CJNE R3,#01H,S+5
    MOV A,#08H
SCANC28: RET

SCANC3:  LCALL SCANC10      ;CAP
    CJNE A,#61H,S+3
    JC SCANC32
    CJNE A,#7BH,S+3
    JNC SCANC32
    SUBB A,#1FH
SCANC32: RET

SCANC4:  LCALL SCANC10      ;CTRL
    ADD A,#80H
    RET

;********** SOUND SUB **********
SOUND:  MOV R5,#0       ;END FLAG
    MOV R4,#80H     ;DELAY CONSTANT
SOUND1:  LCALL SOUNDS
    CJNE R5,#1,SOUND1
    RET

SOUNDS:  MOV DPTR,#58255C   ;OUT 1
    MOVX A,@DPTR
    SETB ACC.5
    MOVX @DPTR,A
    LCALL SOUNDX
    MOV DPTR,#58255C    ;OUT 0
    MOVX A,@DPTR
    CLR ACC.5
    MOVX @DPTR,A
    LCALL SOUNDX
    RET

SOUNDX:  MOV A,R2       ;FREQUENCY DELAY
SOUNDX1: LCALL SOUNDY
    DEC A
    JNZ SOUNDX1
    RET

SOUNDY:  DJNZ R4,SOUNDY1    ;LENGTH COUNT DOWN
    MOV R4,#80H
    DJNZ R3,SOUNDY1
    MOV R5,#1
SOUNDY1: RET
;********** BEEP SUB **********
FBEEP:  MOV R2,#50H     ;DIRECT FUNCTION KEY
    MOV R3,#10H
    LCALL SOUND
    RET

XBEEP:  MOV R2,#40H     ;HEX KEY BEEP
    MOV R3,#10H
    LCALL SOUND
    RET

OBEEP:  MOV R2,#30H     ;OPERATE KEY BEEP
    MOV R3,#10H
    LCALL SOUND
    RET

HBEEP:  MOV R2,#20H     ;SUCCESS BEEP
    MOV R3,#40H
    LCALL SOUND
    MOV R2,#10H
    MOV R3,#20H
    LCALL SOUND
    RET

LBEEP:  MOV R2,#90H     ;WARING BEEP
    MOV R3,#90H
    LCALL SOUND
    RET

RBEEP:  MOV R2,#4       ;READY BEEP
RBEEP1: MOV A,R2
    PUSH ACC        ;-[
    MOV R2,#70H
    MOV R3,#15H
    LCALL SOUND
    MOV R2,#00H
    MOV R3,#18H
    LCALL DMSEC
    POP ACC         ;-]
    MOV R2,A
    DJNZ R2,RBEEP1
    RET
```

```
EBEEP:  MOV R2,#90H        ;ERROR BEEP
        MOV R3,#00H
        LCALL SOUND
        MOV R2,#0C0H
        MOV R3,#00H
        LCALL SOUND
        RET

PBEEP:  MOV R6,#30H        ;POWER-UP BEEP
        MOV R7,#38H
PBEEP2: MOV A,R7
        MOV R2,A
        MOV R3,#02H
        LCALL SOUND
        MOV R2,#00H
        MOV R3,#08H
        LCALL DMSEC
        DEC R7
        DJNZ R6,PBEEP2
        RET
;********** SONG SUB **********
SONG:   CLR A
        MOVC A,@A+DPTR
        INC DPTR
        CJNE A,#0FFH,SONG1
        RET             ;END BY 0FFH

SONG1:  MOV R2,A        ;MEMORY FREE
        CLR A
        MOVC A,@A+DPTR  ;LENGTH
        MOV R3,A
        INC DPTR

        PUSH DPH
        PUSH DPL
        MOV A,R2
        MOV DPTR,#SONGT1
        MOVC A,@A+DPTR
        MOV R2,A        ;FREQUENCY

        MOV A,R3
        MOV DPTR,#SONGT2
        MOVC A,@A+DPTR
        MOV R3,A        ;LENGTH

        LCALL SOUND
        MOV R2,#40H     ;DELAY
SONG4:  MOV R3,#0
        DJNZ R3,$
        DJNZ R2,SONG4
        POP DPL
        POP DPH
        SJMP SONG

SONGT1: DB 000H,0F4H,0E4H,0D9H  ;G,G#,A,A#
        DB 0CCH,0C0H,0B4H,0AAH  ;B,C,C#,D
        DB 0A1H,098H,090H,088H  ;D#,E,F,F#
        DB 080H,078H,072H,06BH  ;G,G#,A,A#
        DB 065H,05FH,05AH,055H  ;B,C,C#,D
        DB 050H,04CH,047H,043H  ;D#,E,F,F#
        DB 03FH,03BH,038H,035H  ;G,G#,A,A#
        DB 032H,02FH,02CH,02AH  ;B,C,C#,D
        DB 027H,025H,023H,021H  ;D#,E,F,F#
        DB 01FH,001H    ;G,S
SONGT2: DB 008H,040H,080H,0C0H  ;0,1,2,3
        DB 000H         ;4
;********** DMSEC SUB **********
DMSEC:  MOV R4,#230     ;1 MSEC LOOP
DMSEC1: NOP
        NOP
        DJNZ R4,DMSEC1
        DJNZ R3,DMSEC
        MOV A,R2
        CJNE A,#0,DMSEC2
        RET
DMSEC2: DEC R2
        SJMP DMSEC
;********** DTSEC SUB **********
DTSEC:  MOV R3,#179
DTSEC1: MOV R4,#0
        DJNZ R4,$
        DJNZ R3,DTSEC1
        DJNZ R2,DTSEC
        RET
;********** DPDEC SUB **********
DPDEC:  XCH A,DPL
        JNZ $+4
        DEC DPH
        DEC A
        XCH A,DPL
        RET

;********** DPADD SUB **********
DPADD:  MOV A,DPL
        ADD A,R3
        MOV DPL,A
        MOV A,DPH
        ADDC A,R2
        MOV DPH,A
        RET
;********** DPSUB SUB **********
DPSUB:  CLR C
        MOV A,DPL
        SUBB A,R3
        MOV DPL,A
        MOV A,DPH
        SUBB A,R2
        MOV DPH,A
        RET
;********** RBYTE SUB **********
RBYTE:  JNB RI,$        ;WAIT FOR RECEIVE OK
        CLR RI
        MOV A,SBUF
        RET
;********** RBYTEH SUB **********
RBYTEH: LCALL RBYTE
        MOV R2,A
        LCALL RBYTE
        MOV R3,A
        LCALL ATOH
        RET
RBYTEHC: LCALL RBYTEH   ;R7=CHECKSUM
        XCH A,R7
        ADD A,R7
```

```
    XCH A,R7
    RET
;********** SBYTE SUB **********
SBYTE:  JNB TI,$        ;WAIT FOR SEND OK
    CLR TI
    MOV SBUF,A
    RET
;********** SBYTEH SUB ***********
SBYTEH:  LCALL HTOA
    MOV A,R2
    LCALL SBYTE
    MOV A,R3
    LCALL SBYTE
    RET
SBYTEHC: XCH A,R7       ;R7=CHECKSUM
    ADD A,R7
    XCH A,R7
    LCALL SBYTEH
    RET
;********** SBLOCK SUB ***********
SBLOCK:  CLR A
    MOVC A,@A+DPTR
    CJNE A,#0,SBLOCK1
    INC DPTR        ;NEXT ADDRESS
    RET             ;EXIT BY 0
SBLOCK1: CJNE A,#0DH,SBLOCK2
    LCALL SLF       ;CR/LF
    INC DPTR        ;NEXT ADDRESS
    RET
SBLOCK2: LCALL SBYTE
    INC DPTR
    SJMP SBLOCK
;********** SLF SUB **********
SLF:   MOV A,#0DH
    LCALL SBYTE
    MOV A,#0AH
    LCALL SBYTE
    RET
;********** HTOA SUB **********
HTOA:  PUSH ACC
    SWAP A
    LCALL HTOAS
    MOV R2,A
    POP ACC
    LCALL HTOAS
    MOV R3,A
    RET
HTOAS:  ANL A,#0FH
    CJNE A,#0AH,$+3
    JNC HTOAS1
    ORL A,#30H
    RET
HTOAS1: SUBB A,#9
    ORL A,#40H
    RET
;********** ATOH SUB **********
ATOH:  MOV A,R2
    LCALL ATOHS
    SWAP A
    MOV R2,A
    MOV A,R3
    LCALL ATOHS
    ORL A,R2
    RET
ATOHS:  CJNE A,#'A',$+3
    JC ATOHS1
    ADD A,#9
ATOHS1: ANL A,#0FH
    RET

;********** ASSEMBLER **********
;USE STTADD = ASCII ADDRESS
;   ENDADD = LAST INPBUF THAT ''
;   MEMADD = OPCODE ADDRESS
ASMT1:  DB '  PROGRAM EDITOR  '
    DB '  :          '
    DB '             '
    DB 'CMM:         '
ASMT2:  DB '  Syntax Error !  '
ASMT3:  DB ' Command Error !  '
ASMT4:  DB '  PROGRAM EDITOR  '

ASM:   MOV DPTR,#8000H
    CLR A
ASM1:  MOVX @DPTR,A
    INC DPTR
    MOV R2,DPH
    CJNE R2,#0A0H,ASM1
    MOV A,#38H
    MOVX @DPTR,A
    INC DPTR
    MOV A,#30H
    MOVX @DPTR,A
    INC DPTR
    MOVX @DPTR,A
    INC DPTR
    MOVX @DPTR,A
    INC DPTR
    MOV A,#20H
ASM2:  MOVX @DPTR,A
    INC DPTR
    MOV R2,DPH
    CJNE R2,#0FCH,ASM2
    MOV STTADD,#0A0H
    MOV STTADD+1,#00H
    MOV MEMADD,#80H
    MOV MEMADD+1,#00H

ASM3:  MOV DPTR,#ASMT1
    LCALL LCDOUT
    LCALL BLACK
    MOV DPH,STTADD
    MOV DPL,STTADD+1

    MOV A,#0C0H     ;WRITE NEW ADDRESS
    LCALL LCDWI
    MOV R4,#4
ASM4:  MOVX A,@DPTR
    LCALL LCDWD
    INC DPTR
    DJNZ R4,ASM4

    MOV A,#94H      ;WRITE NEW PROGRAM
    LCALL LCDWI
```

```
    MOV R4,#20
ASM5:  MOVX A,@DPTR
    LCALL LCDWD
    INC DPTR
    DJNZ R4,ASM5

    MOV A,#0D8H        ;WRITE NEW COMMENT
    LCALL LCDWI
    MOV R4,#16
ASM6:  MOVX A,@DPTR
    LCALL LCDWD
    INC DPTR
    DJNZ R4,ASM6

    MOV R7,#94H
    MOV A,R7
    LCALL LCDWI

ASM7:  LCALL SCAN
    MOV R2,#2
    LCALL DTSEC

    CJNE A,#0H,ASM72
    CJNE R7,#0A8H,$+3
    JNC ASM712
    MOV R2,STTADD      ;UP=PROGRAM
    MOV R3,STTADD+1
    CJNE R2,#0A1H,$+3
    JNC $+7
    CJNE R3,#00H,$+5
    SJMP ASM7
    MOV DPH,STTADD
    MOV DPL,STTADD+1
    MOV R2,#0
    MOV R3,#40
    LCALL DPSUB
    MOV STTADD,DPH
    MOV STTADD+1,DPL
    LJMP ASM3
ASM712: MOV R7,#94H        ;UP=COMMENT
    MOV A,R7
    LCALL LCDWI
    SJMP ASM7

ASM72:  CJNE A,#81H,ASM73    ;DOWN=COMMENT
    MOV R7,#0D8H
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM73:  CJNE A,#7FH,ASM74    ;LEFT
    CJNE R7,#0D8H,ASM732
    MOV R7,#0A7H
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM732: CJNE R7,#94H,$+6
    LJMP ASM7
    DEC R7
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM74:  CJNE A,#7EH,ASM75    ;RIGHT
    CJNE R7,#0A7H,ASM742
    MOV R7,#0D8H
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM742: CJNE R7,#0E7H,$+6
    LJMP ASM7
    INC R7
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM75:  CJNE A,#82H,$+3      ;CHARACTER
    JNC ASM76
    LCALL LCDWD
    CJNE R7,#0A7H,$+5
    SJMP $+5
    CJNE R7,#0E7H,ASM752
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM752: INC R7
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM76:  CJNE A,#82H,ASM77    ;BACK
    CJNE R7,#94H,$+6
    LJMP ASM7
    CJNE R7,#0D8H,$+6
    LJMP ASM7
    MOV A,R7
    MOV R6,A
    CJNE R7,#0A8H,$+3
    JNC ASM762
ASM761: LCALL LCDRD
    MOV R4,A
    DEC R6
    MOV A,R6
    LCALL LCDWI
    MOV A,R4
    LCALL LCDWD
    INC R6
    INC R6
    MOV A,R6
    LCALL LCDWI
    CJNE R6,#0A8H,ASM761
    DEC R6
    MOV A,R6
    LCALL LCDWI
    MOV A,#20H
    LCALL LCDWD
    DEC R7
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM762: LCALL LCDRD
    MOV R4,A
    DEC R6
    MOV A,R6
    LCALL LCDWI
    MOV A,R4
    LCALL LCDWD
    INC R6
```

```
    INC R6
    MOV A,R6
    LCALL LCDWI
    CJNE R6,#0E8H,ASM762
    DEC R6
    MOV A,R6
    LCALL LCDWI
    MOV A,#20H
    LCALL LCDWD
    DEC R7
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM77:  CJNE A,#83H,ASM78    ;DEC
    MOV A,R7
    MOV R6,A
    CJNE A,#0A8H,$+3
    JNC ASM772
    CJNE A,#0A7H,$+5
    SJMP ASM7712
    INC A
    LCALL LCDWI
ASM771: LCALL LCDRD
    MOV R4,A
    MOV A,R6
    LCALL LCDWI
    MOV A,R4
    LCALL LCDWD
    INC R6
    INC R6
    MOV A,R6
    LCALL LCDWI
    DEC R6
    CJNE R6,#0A7H,ASM771
    MOV A,R6
    LCALL LCDWI
ASM7712: MOV A,#20H
    LCALL LCDWD
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM772: CJNE A,#0E7H,$+5
    SJMP ASM7732
    INC A
    LCALL LCDWI
ASM773: LCALL LCDRD
    MOV R4,A
    MOV A,R6
    LCALL LCDWI
    MOV A,R4
    LCALL LCDWD
    INC R6
    INC R6
    MOV A,R6
    LCALL LCDWI
    DEC R6
    CJNE R6,#0E7H,ASM773
    MOV A,R6
    LCALL LCDWI
ASM7732: MOV A,#20H
    LCALL LCDWD
    MOV A,R7
    LCALL LCDWI
    LJMP ASM7
ASM78:  CJNE A,#84H,$+5      ;ENTER
    SJMP $+5
    LJMP ASM7
    MOV DPH,STTADD
    MOV DPL,STTADD+1

    MOV A,#0C0H
    LCALL LCDWI
    MOV R4,#4
ASM781: LCALL LCDRD
    MOVX @DPTR,A
    INC DPTR
    DJNZ R4,ASM781

    MOV A,#94H
    LCALL LCDWI
    MOV R4,#20
ASM782: LCALL LCDRD
    MOVX @DPTR,A
    INC DPTR
    DJNZ R4,ASM782

    MOV A,#0D8H
    LCALL LCDWI
    MOV R4,#16
ASM783: LCALL LCDRD
    MOVX @DPTR,A
    INC DPTR
    DJNZ R4,ASM783

    MOV R0,#INPBUF
    MOV A,#94H
    LCALL LCDWI
    MOV R4,#20
ASM784: LCALL LCDRD
    MOV @R0,A
    INC R0
    DJNZ R4,ASM784
    LCALL ASMC

    MOV R0,#INPBUF
    LCALL TEXT
    CJNE R5,#0,$+6
    LJMP MAIN
    LCALL ASMX
    CJNE A,#0,$+8
    LCALL ASMY
    SJMP $+5
    LJMP ASM7
    LJMP ASM3

;********** ASMX SUB **********
ASMX:   CLR A            ;FIRST CLEAR
    MOV IXDATA,A
    MOV IXDATA+1,A
    MOV IYDATA,A
    MOV IYDATA+1,A
    MOV IZDATA,A
    MOV IZDATA+1,A
```

```
    MOV R2,#4          ;CLEAR OUTPUT
    MOV R1,#TABMEM
ASMX0:  MOV @R1,A
    INC R1
    DJNZ R2,ASMX0

ASMX1:  MOV ENDADD,R0      ;**** MNEMONIC ****
    CJNE R5,#9,$+3
    JC $+5
    LJMP ASMXE         ;CHAR>8
    LCALL ASMXL
    LCALL MTOH
    CJNE A,#0,$+6
    LJMP ASMXE         ;MNEMONIC ERROR
    MOV TABMEM,A       ;CODE

    MOV R0,ENDADD      ;**** OPERAND1 ****
    LCALL TEXT
    CJNE R5,#0,ASMX2
    LJMP ASMX7         ;NO OPERAND
ASMX2:  MOV ENDADD,R0
    CJNE R5,#9,$+3
    JC $+5
    LJMP ASMXE         ;CHAR>8
    LCALL ASMXL
    LCALL OTOH
    CJNE A,#0,$+6
    LJMP ASMX21
    MOV TABMEM+1,A     ;CODE
    SJMP ASMX30
ASMX21: LCALL ASMXH
    CJNE A,#0,$+6
    LJMP ASMXE
    MOV TABMEM+1,A     ;HEX-CODE
    MOV IXDATA,R2      ;HEX
    MOV IXDATA+1,R3

ASMX30: MOV R0,ENDADD      ;**** OPERAND2 ****
    LCALL TEXT
    CJNE R5,#0,ASMX3
    LJMP ASMX7         ;NO OPERAND
ASMX3:  MOV ENDADD,R0
    CJNE R5,#9,$+3
    JC $+5
    LJMP ASMXE         ;CHAR>8
    LCALL ASMXL
    LCALL OTOH
    CJNE A,#0,$+6
    LJMP ASMX31
    MOV TABMEM+2,A     ;CODE
    SJMP  ASMX40
ASMX31: LCALL ASMXH
    CJNE A,#0,$+6
    LJMP ASMXE
    MOV TABMEM+2,A     ;HEX-CODE
    MOV IYDATA,R2      ;HEX
    MOV IYDATA+1,R3

ASMX40: MOV R0,ENDADD      ;**** OPERAND3 ****
    LCALL TEXT
    CJNE R5,#0,ASMX4
    LJMP ASMX7         ;NO OPERAND
ASMX4:  MOV ENDADD,R0
    CJNE R5,#9,$+3
    JNC ASMXE
    LCALL ASMXL
    LCALL OTOH
    CJNE A,#0,$+6
    LJMP ASMX41
    MOV TABMEM+3,A     ;CODE
    SJMP ASMX5
ASMX41: LCALL ASMXH
    CJNE A,#0,$+6
    LJMP ASMXE
    MOV TABMEM+3,A     ;HEX-CODE
    MOV IZDATA,R2      ;HEX
    MOV IZDATA+1,R3

ASMX5:  MOV R0,ENDADD      ;CHECK END
    LCALL TEXT
    CJNE R5,#0,ASMXE

ASMX7:  CLR A
    RET

ASMXE:  MOV DPTR,#ASMT2    ;ASM ERROR
    LCALL LCDOUT1
    LCALL EBEEP
    MOV R2,#10
    LCALL DTSEC
    MOV DPTR,#ASMT4
    LCALL LCDOUT1
    MOV R7,#94H
    MOV A,R7
    LCALL LCDWI
    MOV A,#1
    RET

;********** ASMY SUB **********
ASMY:   MOV R0,#TABMEM     ;LOAD TABMEM TO ASMBUF
    MOV R1,#ASMBUF
    MOV R2,#4
ASMY1:  MOV A,@R0
    MOV @R1,A
    INC R0
    INC R1
    DJNZ R2,ASMY1

    MOV A,ASMBUF
    CJNE A,#45,ASMY12
    MOV A,ASMBUF+1
    CJNE A,#1,$+5
    SJMP $+5
    LJMP ASMY20
    MOV MEMADD,IXDATA
    MOV MEMADD+1,IXDATA+1
    MOV DPH,STTADD
    MOV DPL,STTADD+1
    MOV R2,#0
    MOV R3,#40
    LCALL DPADD
    MOV STTADD,DPH
    MOV STTADD+1,DPL
    MOV A,#0C0H
```

```
        LCALL LCDWI
        MOV A,IXDATA
        LCALL HTOA
        MOV A,R2
        MOVX @DPTR,A
        INC DPTR
        MOV A,R3
        MOVX @DPTR,A
        INC DPTR
        MOV A,IXDATA+1
        LCALL HTOA
        MOV A,R2
        MOVX @DPTR,A
        INC DPTR
        MOV A,R3
        MOVX @DPTR,A
        RET

ASMY12: CJNE A,#46,ASMY20
        MOV A,ASMBUF+1
        CJNE A,#1,ASMY20
        MOV ASMBUF+3,#1
        MOV DPH,MEMADD
        MOV DPL,MEMADD+1
        MOV A,IXDATA+1
        MOVX @DPTR,A
        INC DPTR
        MOV MEMADD,DPH
        MOV MEMADD+1,DPL
        MOV DPH,STTADD
        MOV DPL,STTADD+1
        MOV R2,#0
        MOV R3,#40
        LCALL DPADD
        MOV STTADD,DPH
        MOV STTADD+1,DPL
        MOV A,#0C0H
        LCALL LCDWI
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV DPH,A
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV DPL,A
        INC DPTR
        MOV A,DPH
        LCALL HTOA
        MOV R4,DPL
        MOV DPH,STTADD
        MOV DPL,STTADD+1
        MOV A,R2
        MOVX @DPTR,A
        INC DPTR
        MOV A,R3
        MOVX @DPTR,A
        MOV A,R4
        LCALL HTOA
        INC DPTR
        MOV A,R2
        MOVX @DPTR,A
        INC DPTR
        MOV A,R3
        MOVX @DPTR,A
        RET

ASMY20: MOV DPTR,#MCS51T    ;SEARCH TABLE
        MOV R7,#0           ;OPCODE COUNT (256)
ASMY2:  MOV R2,#4
        MOV R3,#0
        MOV R1,#ASMBUF
ASMY21: MOV A,R3            ;COMPARE
        MOVC A,@A+DPTR
        MOV B,A
        MOV A,@R1
        CJNE A,B,ASMY25
        INC R1
        INC R3
        DJNZ R2,ASMY21
        SJMP ASMY3          :=
ASMY25: MOV R2,#0           ;NEXT COMPARE
        MOV R3,#5
        LCALL DPADD
        INC R7
        CJNE R7,#0,ASMY2
        LJMP ASMYF          ;NOT FOUND

ASMY3:  MOV ASMBUF,R7       ;FRIST OPCODE
        MOV A,#4
        MOVC A,@A+DPTR
        PUSH ACC
        ANL A,#0FH
        MOV ASMBUF+3,A      ;NO. OF BYTE
        POP ACC
        SWAP A
        ANL A,#0FH          ;A=PATTERN

        CJNE A,#1,ASMY42    ;**** PAT1 ****
        MOV A,IXDATA        ;11 BIT ADDRESS
        CJNE A,#80H,$+3
        JNC $+5
        LJMP ASMYE
        CJNE A,#0A0H,$+3
        JC $+5
        LJMP ASMYE
        ANL A,#0F8H
        MOV B,A
        MOV A,MEMADD
        ANL A,#0F8H
        CJNE A,B,$+5        ;OUT OF RANGE
        SJMP $+5
        LJMP ASMYF
        MOV A,IXDATA        ;A10 A9 A8
        SWAP A
        RL A
        ANL A,#0E0H
        MOV B,A
        CLR A               ;CHECK ACALL OR AJMP
        MOVC A,@A+DPTR
```

```
        CJNE A,#1,ASMY32
        MOV A,B          ;ACALL
        ORL A,#11H
        SJMP ASMY34
ASMY32: MOV A,B          ;AJMP
        ORL A,#01H
ASMY34: MOV ASMBUF,A
        MOV A,IXDATA+1   ;A7 - A0
        MOV ASMBUF+1,A
        LJMP ASMY7

ASMY42: CJNE A,#2,ASMY43 ;**** PAT2 ****
        MOV A,IXDATA
        CJNE A,#80H,S+3
        JNC S+5
        LJMP ASMYE
        CJNE A,#0A0H,S+3
        JC S+5
        LJMP ASMYE

        MOV ASMBUF+1,A
        MOV A,IXDATA+1
        MOV ASMBUF+2,A
        LJMP ASMY7

ASMY43: CJNE A,#3,ASMY44 ;**** PAT3 ****
        MOV A,#1
        MOVC A,@A+DPTR
        CJNE A,#1,ASMY431
        MOV A,IXDATA     ;OPERAND1
        CJNE A,#0,ASMYF
        MOV A,IXDATA+1
        MOV ASMBUF+1,A
        LJMP ASMY7
ASMY431: MOV A,IYDATA    ;OPERAND2
        CJNE A,#0,ASMYF
        MOV A,IYDATA+1
        MOV ASMBUF+1,A
        LJMP ASMY7

ASMY44: CJNE A,#4,ASMY45 ;**** PAT4 ****
        MOV A,IYDATA
        CJNE A,#80H,S+3
        JNC S+5
        LJMP ASMYE
        CJNE A,#0A0H,S+3
        JC S+5
        LJMP ASMYE

        MOV A,IXDATA
        CJNE A,#0,ASMYF
        MOV A,IXDATA+1
        MOV ASMBUF+1,A
        MOV DPH,IYDATA
        MOV DPL,IYDATA+1
        LCALL CREL
        MOV ASMBUF+2,A
        LJMP ASMY7

ASMY45: CJNE A,#5,ASMY46 ;**** PAT5 ****
        MOV A,#1
        MOVC A,@A+DPTR
        CJNE A,#1,ASMY451
        MOV A,IXDATA     ;OPERAND1
        CJNE A,#0,ASMYF
        MOV A,IXDATA+1
        MOV ASMBUF+1,A
        LJMP ASMY7

ASMYF:  LJMP ASMYE

ASMY451: MOV A,IYDATA    ;OPERAND2
        CJNE A,#0,ASMYF
        MOV A,IYDATA+1
        MOV ASMBUF+1,A
        LJMP ASMY7

ASMY46: CJNE A,#6,ASMY47 ;**** PAT6 ****
        SJMP ASMY472
ASMY47: CJNE A,#7,ASMY48 ;**** PAT7 ****
ASMY472: MOV A,IXDATA
        CJNE A,#0,ASMYF
        MOV A,IXDATA+1
        MOV ASMBUF+1,A
        MOV A,IYDATA
        CJNE A,#0,ASMYF
        MOV A,IYDATA+1
        MOV ASMBUF+2,A
        LJMP ASMY7

ASMY48: CJNE A,#8,ASMY49 ;**** PAT8 ****
        MOV A,IZDATA
        CJNE A,#80H,S+3
        JNC S+5
        LJMP ASMYE
        CJNE A,#0A0H,S+3
        JC S+5
        LJMP ASMYE

        MOV A,#1
        MOVC A,@A+DPTR
        CJNE A,#1,ASMY481
        MOV A,IXDATA     ;OPERAND1
        CJNE A,#0,ASMYF
        MOV A,IXDATA+1
        MOV ASMBUF+1,A
        MOV DPH,IYDATA
        MOV DPL,IYDATA+1
        LCALL CREL
        MOV ASMBUF+2,A
        LJMP ASMY7
ASMY481: MOV A,IYDATA    ;OPERAND2
        CJNE A,#0,ASMYF
        MOV A,IYDATA+1
        MOV ASMBUF+1,A
        MOV DPH,IZDATA
        MOV DPL,IZDATA+1
        LCALL CREL
        MOV ASMBUF+2,A
        LJMP ASMY7

ASMY49: CJNE A,#9,ASMY4A ;**** PAT9 ****
        MOV A,IYDATA
        CJNE A,#0,ASMYF
```

```
        MOV A,IYDATA+1
        MOV ASMBUF+1,A
        LJMP ASMY7

ASMY4A: CJNE A,#10,ASMY4B   ;**** PAT10 ****
        MOV A,IZDATA
        CJNE A,#80H,$+3
        JNC $+5
        LJMP ASMYE
        CJNE A,#0A0H,$+3
        JC $+5
        LJMP ASMYE

        MOV A,IYDATA
        CJNE A,#0,$+5
        SJMP $+5
        LJMP ASMYE
        MOV A,IYDATA+1
        MOV ASMBUF+1,A
        MOV DPH,IZDATA
        MOV DPL,IZDATA+1
        LCALL CREL
        MOV ASMBUF+2,A
        LJMP ASMY7

ASMY4B: CJNE A,#11,ASMY4C   ;**** PAT11 ****
        MOV A,IYDATA
        MOV ASMBUF+1,A
        MOV A,IYDATA+1
        MOV ASMBUF+2,A
        LJMP ASMY7

ASMY4C: CJNE A,#12,ASMY7    ;**** PAT12 ****
        MOV A,IXDATA
        CJNE A,#80H,$+3
        JNC $+5
        LJMP ASMYE
        CJNE A,#0A0H,$+3
        JC $+5
        LJMP ASMYE

        MOV A,#1
        MOVC A,@A+DPTR
        CJNE A,#1,ASMY4C1
        MOV DPH,IXDATA
        MOV DPL,IXDATA+1
        LCALL CREL
        MOV ASMBUF+1,A
        LJMP ASMY7
ASMY4C1: MOV DPH,IYDATA
        MOV DPL,IYDATA+1
        LCALL CREL
        MOV ASMBUF+1,A

ASMY7:  MOV DPH,MEMADD      ;LOAD TO STTADD
        MOV DPL,MEMADD+1
        MOV R1,#ASMBUF
        MOV R7,ASMBUF+3
ASMY71: MOV A,@R1
        MOVX @DPTR,A
        INC DPTR
        INC R1
        DJNZ R7,ASMY71
        MOV MEMADD,DPH
        MOV MEMADD+1,DPL
        MOV DPH,STTADD
        MOV DPL,STTADD+1
        MOV R2,#0
        MOV R3,#40
        LCALL DPADD
        MOV STTADD,DPH
        MOV STTADD+1,DPL
        MOV A,#0C0H
        LCALL LCDWI
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV DPH,A
        LCALL LCDRD
        MOV R2,A
        LCALL LCDRD
        MOV R3,A
        LCALL ATOH
        MOV DPL,A
        MOV R2,#0
        MOV R3,ASMBUF+3
        LCALL DPADD
        MOV A,DPH
        MOV R4,DPL
        LCALL HTOA
        MOV DPH,STTADD
        MOV DPL,STTADD+1
        MOV A,R2
        MOVX @DPTR,A
        INC DPTR
        MOV A,R3
        MOVX @DPTR,A
        MOV A,R4
        LCALL HTOA
        INC DPTR
        MOV A,R2
        MOVX @DPTR,A
        INC DPTR
        MOV A,R3
        MOVX @DPTR,A
        RET

ASMYE:  MOV DPTR,#ASMT3     ;ASM ERROR
        LCALL LCDOUT1
        LCALL EBEEP
        MOV R2,#10
        LCALL DTSEC
        MOV DPTR,#ASMT4
        LCALL LCDOUT1
        MOV R7,#94H
        MOV A,R7
        LCALL LCDWI
        RET

;********** CREL SUB **********
CREL:   MOV R2,MEMADD
        MOV R3,MEMADD+1
```

```
    MOV A,ASMBUF+3
    DEC A
    ADD A,R3
    MOV R3,A
    MOV A,#0
    ADDC A,R2
    MOV R2,A        ;R2,R3 = rr ADDRESS

    LCALL DPCOM
    JZ CREL4
    JC CREL4

    LCALL DPSUB     ;JUMP TO HIGH
    LCALL DPDEC
    MOV A,DPH       ;CHECK OVER RANGE
    CJNE A,#0,CRELE
    MOV A,DPL
    CJNE A,#80H,$+3
    JNC CRELE
    RET             ;A=OUTPUT

                    ;JUMP TO LOW
CREL4:  XCH A,DPH       ;SWAP DPTR - R2,R3
    XCH A,R2
    XCH A,DPH
    XCH A,DPL
    XCH A,R3
    XCH A,DPL
    LCALL DPSUB     ;CHECK OVER RANGE
    MOV A,DPH
    CJNE A,#0,CRELE
    MOV A,DPL
    CJNE A,#80H,$+3
    JNC CRELE
    CPL A
    RET             ;A=OUTPUT

CRELE:  LCALL ASMYE     ;RANGE ERROR
    POP ACC
    POP ACC         ;CLEAR RET
    RET

;********** ASMXH SUB **********
ASMXH:  MOV A,ASMBUF    ;CHECK # AND /
    CJNE A,#'#',ASMXH1
    MOV R7,#2
    SJMP ASMXH4
ASMXH1: CJNE A,#'/',ASMXH2
    MOV R7,#3
    JMP ASMXH4
ASMXH2: MOV R7,#1
    JMP ASMXH5

ASMXH4: MOV R0,#ASMBUF  ;SHIFT FIRST BYTE (#/)
    MOV R1,#ASMBUF+1
    MOV R2,#7
ASMXH41: MOV A,@R1
    MOV @R0,A
    INC R0
    INC R1
    DJNZ R2,ASMXH41
    MOV ASMBUF+7,#' '

ASMXH5: MOV A,ASMBUF    ;CHECK HEX
    CJNE A,#'0',$+3
    JC ASMXH7
    CJNE A,#'9'+1,$+3
    JNC ASMXH7

    MOV R0,#ASMBUF  ;IS REAL-HEX
    MOV R2,#0
    MOV R3,#0
    MOV R4,#0       ;DIGIT COUNT
    MOV R5,#8
ASMXH6: MOV A,@R0       ;BY PASS FIRST ZERO
    CJNE A,#'0',ASMXH61
    INC R0
    DJNZ R5,ASMXH6
    SJMP ASMXH65    ;IS ZERO

ASMXH61: MOV A,@R0
    CJNE A,#' ',$+6
    LJMP ASMXH65
    CJNE A,#'H',$+6
    LJMP ASMXH65
    CJNE A,#'0',$+3
    JC ASMXH6E      ;<0
    CJNE A,#'F'+1,$+3
    JNC ASMXH6E     ;>F
    CJNE A,#'A',$+3
    JNC ASMXH62     ;IS A-F
    CJNE A,#'9'+1,$+3
    JNC ASMXH6E     ;>9 <A
ASMXH62: LCALL ATOHS    ;ASCII TO HEX
    LCALL SHIFT4
    INC R0
    INC R4
    DJNZ R5,ASMXH61

ASMXH65: CJNE R4,#5,$+3
    JNC ASMXH6E
    MOV A,R7
    RET             ;EXIT REAL-HEX
ASMXH6E: CLR A
    RET             ;EXIT HEX-ERROR

ASMXH7: LCALL NTOH      ;CHECK SFR,BIT NAME
    CJNE A,#0,$+6
    LJMP ASMXH72
    MOV R2,#0
    MOV R3,A
    MOV A,R7
    RET             ;EXIT SFR-NAME
ASMXH72: CALL BTOH
    CJNE A,#0,$+6
    LJMP ASMXH73
    MOV R2,#0
    MOV R3,A
    MOV A,R7
    RET             ;EXIT BIT-NAME
ASMXH73: CLR A
    RET             ;EXIT TEXT-ERROR
;********** ASMXC SUB **********
```

```
ASMXC:  MOV R1,#ASMBUF      ;CLEAR ASMBUF
        MOV R2,#8
ASMXC1: MOV @R1,#' '
        INC R1
        DJNZ R2,ASMXC1
        RET

;********** ASMXL SUB **********
ASMXL:  LCALL ASMXC
        MOV A,R0
        CLR C
        SUBB A,R5
        MOV R0,A
        MOV R1,#ASMBUF      ;LOAD INPBUF TO ASMBUF
ASMXL1: MOV A,@R0
        MOV @R1,A
        INC R1
        INC R0
        DJNZ R5,ASMXL1
        RET

; ********** ASMC SUB **********
ASMC:   MOV R0,#INPBUF
        MOV R7,#20
ASMC1:  MOV A,@R0
        CJNE A,#',',ASMC2
        MOV A,#' '
        MOV @R0,A
        SJMP ASMC3
ASMC2:  CJNE A,#61H,S+3
        JC ASMC3
        CJNE A,#7BH,S+3
        JNC ASMC3
        SUBB A,#31
        MOV @R0,A
ASMC3:  INC R0
        DJNZ R7,ASMC1
        RET

;********** MTOH SUB **********
MTOH:   MOV DPTR,#MTOHT
        MOV R4,#1           ;OUTPUT

MTOH1:  MOV R2,#5
        MOV R3,#0
        MOV R1,#ASMBUF
MTOH2:  MOV A,R3            ;COMPARE 5 BYTE
        MOVC A,@A+DPTR
        CLR C
        SUBB A,@R1
        JNZ MTOH3
        INC R1
        INC R3
        DJNZ R2,MTOH2
        MOV A,R4
        RET             ;EXIT

MTOH3:  MOV R2,#0
        MOV R3,#5
        LCALL DPADD
        INC R4
        CJNE R4,#46+1,MTOH1 ;NEXT COMPARE
        CLR A           ;ERROR
        RET

MTOHT:  DB "ACALL"
        DB "ADD  "
        DB "ADDC "
        DB "AJMP "
        DB "ANL  "
        DB "CJNE "
        DB "CLR  "
        DB "CPL  "
        DB "DA   "
        DB "DEC  "
        DB "DIV  "
        DB "DJNZ "
        DB "INC  "
        DB "JB   "
        DB "JBC  "
        DB "JC   "
        DB "JMP  "
        DB "JNB  "
        DB "JNC  "
        DB "JNZ  "
        DB "JZ   "
        DB "LCALL"
        DB "LJMP "
        DB "MOV  "
        DB "MOVC "
        DB "MOVX "
        DB "MUL  "
        DB "NOP  "
        DB "ORL  "
        DB "POP  "
        DB "PUSH "
        DB "RET  "
        DB "RETI "
        DB "RL   "
        DB "RLC  "
        DB "RR   "
        DB "RRC  "
        DB "SETB "
        DB "SJMP "
        DB "SUBB "
        DB "SWAP "
        DB "XCH  "
        DB "XCHD "
        DB "XRL  "
        DB "ORG  "
        DB "DB   "

;********** OTOH SUB **********
OTOH:   MOV DPTR,#OTOHT
        MOV R4,#4           ;OUTPUT

OTOH1:  MOV R2,#7
        MOV R3,#0
        MOV R1,#ASMBUF
OTOH2:  MOV A,R3            ;COMPARE 7 BYTE
        MOVC A,@A+DPTR
        CLR C
        SUBB A,@R1
        JNZ OTOH3
```

```
        INC R1
        INC R3
        DJNZ R2,OTOH2
        MOV A,R4
        RET             ;EXIT

OTOH3:  MOV R2,#0
        MOV R3,#7
        LCALL DPADD
        INC R4
        CJNE R4,#20+1,OTOH1   ;NEXT COMPARE
        CLR A           ;ERROR
        RET

OTOHT:  DB "@A+DPTR"
        DB "@A+PC  "
        DB "@DPTR  "
        DB "@R0    "
        DB "@R1    "
        DB "A      "
        DB "AB     "
        DB "C      "
        DB "DPTR   "
        DB "R0     "
        DB "R1     "
        DB "R2     "
        DB "R3     "
        DB "R4     "
        DB "R5     "
        DB "R6     "
        DB "R7     "

;********** HTON SUB **********
HTON:   MOV R7,#26
        MOV B,R0
        MOV DPTR,#NTOHT
HTON1:  CLR A
        MOVC A,@A+DPTR
        CJNE A,B,HTON2
        INC DPTR
        MOV A,#1
        RET
HTON2:  MOV R2,#0
        MOV R3,#7
        LCALL DPADD
        DJNZ R7,HTON1
        CLR A
        RET

;********** NTOH SUB **********
NTOH:   MOV DPTR,#NTOHT+1
        MOV R4,#26

NTOH1:  MOV R2,#6
        MOV R3,#0
        MOV R1,#ASMBUF
NTOH2:  MOV A,R3        ;COMPARE 5 BYTE
        MOVC A,@A+DPTR
        CLR C
        SUBB A,@R1
        JNZ NTOH3
        INC R1
        INC R3
        DJNZ R2,NTOH2

        LCALL DPDEC     ;FOUND
        CLR A
        MOVC A,@A+DPTR
        RET             ;EXIT

NTOH3:  MOV R2,#0
        MOV R3,#7
        LCALL DPADD
        DJNZ R4,NTOH1   ;NEXT COMPARE
        CLR A           ;ERROR (NOT FOUND)
        RET

NTOHT:  DB 80H,'P0  '
        DB 81H,'SP  '
        DB 82H,'DPL '
        DB 83H,'DPH '
        DB 87H,'PCON '
        DB 88H,'TCON '
        DB 89H,'TMOD '
        DB 8AH,'TL0 '
        DB 8BH,'TL1 '
        DB 8CH,'TH0 '
        DB 8DH,'TH1 '
        DB 90H,'P1  '
        DB 98H,'SCON '
        DB 99H,'SBUF '
        DB 0A0H,'P2  '
        DB 0A8H,'IE  '
        DB 0B0H,'P3  '
        DB 0B8H,'IP  '
        DB 0C8H,'T2CON '
        DB 0CAH,'RCAP2L'
        DB 0CBH,'RCAP2H'
        DB 0CCH,'TL2 '
        DB 0CDH,'TH2 '
        DB 0D0H,'PSW '
        DB 0E0H,'ACC '
        DB 0F0H,'B   '

;********** BTOH SUB **********
BTOH:   MOV DPTR,#BTOHT
        MOV R4,#80H     ;OUTPUT

BTOH1:  MOV R2,#5
        MOV R3,#0
        MOV R1,#ASMBUF
BTOH2:  MOV A,R3        ;COMPARE 5 BYTE
        MOVC A,@A+DPTR
        CLR C
        SUBB A,@R1
        JNZ BTOH3
        INC R1
        INC R3
        DJNZ R2,BTOH2
        MOV A,R4
        RET             ;EXIT

BTOH3:  MOV R2,#0
        MOV R3,#5
```

```
        LCALL DPADD
        INC R4
    BTOH4:  CJNE R4,#0F7H+1,BTOH1  ;NEXT COMPARE
        CLR A           :ERROR
        RET

    BTOHT:  DB "P0.0 "          ;P0 (80)
        DB "P0.1 "
        DB "P0.2 "
        DB "P0.3 "
        DB "P0.4 "
        DB "P0.5 "
        DB "P0.6 "
        DB "P0.7 "
        DB "IT0  "          ;TCON (88)
        DB "IE0  "
        DB "IT1  "
        DB "IE1  "
        DB "TR0  "
        DB "TF0  "
        DB "TR1  "
        DB "TF1  "
        DB "P1.0 "          ;P1 (90)
        DB "P1.1 "
        DB "P1.2 "
        DB "P1.3 "
        DB "P1.4 "
        DB "P1.5 "
        DB "P1.6 "
        DB "P1.7 "
        DB "RI   "          ;SCON (98)
        DB "TI   "
        DB "RB8  "
        DB "TB8  "
        DB "REN  "
        DB "SM2  "
        DB "SM1  "
        DB "SM0  "
        DB "P2.0 "          ;P2 (A0)
        DB "P2.1 "
        DB "P2.2 "
        DB "P2.3 "
        DB "P2.4 "
        DB "P2.5 "
        DB "P2.6 "
        DB "P2.7 "
        DB "EX0  "          ;IE (A8)
        DB "ET0  "
        DB "EX1  "
        DB "ET1  "
        DB "ES   "
        DB "ET2  "
        DB "     "
        DB "EA   "
        DB "RXD  "          ;P3 (B0)
        DB "TXD  "
        DB "INT0 "
        DB "INT1 "
        DB "T0   "
        DB "T1   "
        DB "WR   "
        DB "RD   "
        DB "PX0  "          ;IP (B8)
        DB "PT0  "
        DB "PX1  "
        DB "PT1  "
        DB "PS   "
        DB "PT2  "
        DB "     "
        DB "     "
        DB "     "          ; (C0)
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "TF2  "          ; (C8)
        DB "EXF2 "
        DB "RCLK "
        DB "TLCK "
        DB "EXEN2"
        DB "TR2  "
        DB "CT2  "
        DB "CPRL2"
        DB "P    "          ;PSW (D0)
        DB "     "
        DB "OV   "
        DB "RS0  "
        DB "RS1  "
        DB "F0   "
        DB "AC   "
        DB "CY   "
        DB "     "          ; (D8)
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "ACC.0"          ;ACC (E0)
        DB "ACC.1"
        DB "ACC.2"
        DB "ACC.3"
        DB "ACC.4"
        DB "ACC.5"
        DB "ACC.6"
        DB "ACC.7"
        DB "     "          ; (E8)
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "     "
        DB "B.0  "          ;B (F0)
        DB "B.1  "
        DB "B.2  "
        DB "B.3  "
        DB "B.4  "
        DB "B.5  "
```

```
        DB "B.6 "
        DB "B.7 "
        DB "   "          ; (F8)
        DB "   "
        DB "   "
        DB "   "
        DB "   "
        DB "   "
        DB "   "
        DB "   "

;********** MCS-51 TABLE **********
;MNEMONIC OPERAND1 OPERAND2 OPERAND3 PATTERN,BYTE

MCS51T: DB 1CH,00H,00H,00H,01H  ;00
        DB 04H,01H,00H,00H,12H  ;01
        DB 17H,01H,00H,00H,23H  ;02
        DB 24H,09H,00H,00H,01H  ;03
        DB 0DH,09H,00H,00H,01H  ;04
        DB 0DH,01H,00H,00H,52H  ;05
        DB 0DH,07H,00H,00H,01H  ;06
        DB 0DH,08H,00H,00H,01H  ;07
        DB 0DH,0DH,00H,00H,01H  ;08
        DB 0DH,0EH,00H,00H,01H  ;09
        DB 0DH,0FH,00H,00H,01H  ;0A
        DB 0DH,10H,00H,00H,01H  ;0B
        DB 0DH,11H,00H,00H,01H  ;0C
        DB 0DH,12H,00H,00H,01H  ;0D
        DB 0DH,13H,00H,00H,01H  ;0E
        DB 0DH,14H,00H,00H,01H  ;0F
        DB 0FH,01H,01H,00H,43H  ;10
        DB 01H,01H,00H,00H,12H  ;11
        DB 16H,01H,00H,00H,23H  ;12
        DB 25H,09H,00H,00H,01H  ;13
        DB 0AH,09H,00H,00H,01H  ;14
        DB 0AH,01H,00H,00H,52H  ;15
        DB 0AH,07H,00H,00H,01H  ;16
        DB 0AH,08H,00H,00H,01H  ;17
        DB 0AH,0DH,00H,00H,01H  ;18
        DB 0AH,0EH,00H,00H,01H  ;19
        DB 0AH,0FH,00H,00H,01H  ;1A
        DB 0AH,10H,00H,00H,01H  ;1B
        DB 0AH,11H,00H,00H,01H  ;1C
        DB 0AH,12H,00H,00H,01H  ;1D
        DB 0AH,13H,00H,00H,01H  ;1E
        DB 0AH,14H,00H,00H,01H  ;1F
        DB 0EH,01H,01H,00H,43H  ;20
        DB 04H,01H,00H,00H,12H  ;21
        DB 20H,00H,00H,00H,01H  ;22
        DB 22H,09H,00H,00H,01H  ;23
        DB 02H,09H,02H,00H,92H  ;24
        DB 02H,09H,01H,00H,52H  ;25
        DB 02H,09H,07H,00H,01H  ;26
        DB 02H,09H,08H,00H,01H  ;27
        DB 02H,09H,0DH,00H,01H  ;28
        DB 02H,09H,0EH,00H,01H  ;29
        DB 02H,09H,0FH,00H,01H  ;2A
        DB 02H,09H,10H,00H,01H  ;2B
        DB 02H,09H,11H,00H,01H  ;2C
        DB 02H,09H,12H,00H,01H  ;2D
        DB 02H,09H,13H,00H,01H  ;2E
        DB 02H,09H,14H,00H,01H  ;2F
        DB 12H,01H,01H,00H,43H  ;30
        DB 01H,01H,00H,00H,12H  ;31
        DB 21H,00H,00H,00H,01H  ;32
        DB 23H,09H,00H,00H,01H  ;33
        DB 03H,09H,02H,00H,92H  ;34
        DB 03H,09H,01H,00H,52H  ;35
        DB 03H,09H,07H,00H,01H  ;36
        DB 03H,09H,08H,00H,01H  ;37
        DB 03H,09H,0DH,00H,01H  ;38
        DB 03H,09H,0EH,00H,01H  ;39
        DB 03H,09H,0FH,00H,01H  ;3A
        DB 03H,09H,10H,00H,01H  ;3B
        DB 03H,09H,11H,00H,01H  ;3C
        DB 03H,09H,12H,00H,01H  ;3D
        DB 03H,09H,13H,00H,01H  ;3E
        DB 03H,09H,14H,00H,01H  ;3F
        DB 10H,01H,00H,00H,0C2H ;40
        DB 04H,01H,00H,00H,12H  ;41
        DB 1DH,01H,09H,00H,52H  ;42
        DB 1DH,01H,02H,00H,73H  ;43
        DB 1DH,09H,02H,00H,92H  ;44
        DB 1DH,09H,01H,00H,52H  ;45
        DB 1DH,09H,07H,00H,01H  ;46
        DB 1DH,09H,08H,00H,01H  ;47
        DB 1DH,09H,0DH,00H,01H  ;48
        DB 1DH,09H,0EH,00H,01H  ;49
        DB 1DH,09H,0FH,00H,01H  ;4A
        DB 1DH,09H,10H,00H,01H  ;4B
        DB 1DH,09H,11H,00H,01H  ;4C
        DB 1DH,09H,12H,00H,01H  ;4D
        DB 1DH,09H,13H,00H,01H  ;4E
        DB 1DH,09H,14H,00H,01H  ;4F
        DB 13H,01H,00H,00H,0C2H ;50
        DB 01H,01H,00H,00H,12H  ;51
        DB 05H,01H,09H,00H,52H  ;52
        DB 05H,01H,02H,00H,73H  ;53
        DB 05H,09H,02H,00H,92H  ;54
        DB 05H,09H,01H,00H,52H  ;55
        DB 05H,09H,07H,00H,01H  ;56
        DB 05H,09H,08H,00H,01H  ;57
        DB 05H,09H,0DH,00H,01H  ;58
        DB 05H,09H,0EH,00H,01H  ;59
        DB 05H,09H,0FH,00H,01H  ;5A
        DB 05H,09H,10H,00H,01H  ;5B
        DB 05H,09H,11H,00H,01H  ;5C
        DB 05H,09H,12H,00H,01H  ;5D
        DB 05H,09H,13H,00H,01H  ;5E
        DB 05H,09H,14H,00H,01H  ;5F
        DB 15H,01H,00H,00H,0C2H ;60
        DB 04H,01H,00H,00H,12H  ;61
        DB 2CH,01H,09H,00H,52H  ;62
        DB 2CH,01H,02H,00H,73H  ;63
        DB 2CH,09H,02H,00H,92H  ;64
        DB 2CH,09H,01H,00H,52H  ;65
        DB 2CH,09H,07H,00H,01H  ;66
        DB 2CH,09H,08H,00H,01H  ;67
        DB 2CH,09H,0DH,00H,01H  ;68
        DB 2CH,09H,0EH,00H,01H  ;69
        DB 2CH,09H,0FH,00H,01H  ;6A
        DB 2CH,09H,10H,00H,01H  ;6B
        DB 2CH,09H,11H,00H,01H  ;6C
        DB 2CH,09H,12H,00H,01H  ;6D
```

```
DB 2CH,09H,13H,00H,01H  ;6E
DB 2CH,09H,14H,00H,01H  ;6F
DB 14H,01H,00H,00H,0C2H  ;70
DB 01H,01H,00H,00H,12H  ;71
DB 1DH,0BH,01H,00H,32H  ;72
DB 11H,04H,00H,00H,01H  ;73
DB 18H,09H,02H,00H,92H  ;74
DB 18H,01H,02H,00H,73H  ;75
DB 18H,07H,02H,00H,92H  ;76
DB 18H,08H,02H,00H,92H  ;77
DB 18H,0DH,02H,00H,92H  ;78
DB 18H,0EH,02H,00H,92H  ;79
DB 18H,0FH,02H,00H,92H  ;7A
DB 18H,10H,02H,00H,92H  ;7B
DB 18H,11H,02H,00H,92H  ;7C
DB 18H,12H,02H,00H,92H  ;7D
DB 18H,13H,02H,00H,92H  ;7E
DB 18H,14H,02H,00H,92H  ;7F
DB 27H,01H,00H,00H,0C2H  ;80
DB 04H,01H,00H,00H,12H  ;81
DB 05H,0BH,01H,00H,32H  ;82
DB 19H,09H,05H,00H,01H  ;83
DB 0BH,0AH,00H,00H,01H  ;84
DB 18H,01H,01H,00H,63H  ;85
DB 18H,01H,07H,00H,52H  ;86
DB 18H,01H,08H,00H,52H  ;87
DB 18H,01H,0DH,00H,52H  ;88
DB 18H,01H,0EH,00H,52H  ;89
DB 18H,01H,0FH,00H,52H  ;8A
DB 18H,01H,10H,00H,52H  ;8B
DB 18H,01H,11H,00H,52H  ;8C
DB 18H,01H,12H,00H,52H  ;8D
DB 18H,01H,13H,00H,52H  ;8E
DB 18H,01H,14H,00H,52H  ;8F
DB 18H,0CH,02H,00H,0B3H  ;90
DB 01H,01H,00H,00H,12H  ;91
DB 18H,01H,0BH,00H,32H  ;92
DB 19H,09H,04H,00H,01H  ;93
DB 28H,09H,02H,00H,92H  ;94
DB 28H,09H,01H,00H,52H  ;95
DB 28H,09H,07H,00H,01H  ;96
DB 28H,09H,08H,00H,01H  ;97
DB 28H,09H,0DH,00H,01H  ;98
DB 28H,09H,0EH,00H,01H  ;99
DB 28H,09H,0FH,00H,01H  ;9A
DB 28H,09H,10H,00H,01H  ;9B
DB 28H,09H,11H,00H,01H  ;9C
DB 28H,09H,12H,00H,01H  ;9D
DB 28H,09H,13H,00H,01H  ;9E
DB 28H,09H,14H,00H,01H  ;9F
DB 1DH,0BH,03H,00H,32H  ;A0
DB 04H,01H,00H,00H,12H  ;A1
DB 18H,0BH,01H,00H,32H  ;A2
DB 0DH,0CH,00H,00H,01H  ;A3
DB 1BH,0AH,00H,00H,01H  ;A4
DB 00H,00H,00H,00H,00H  ;A5 RESERVE
DB 18H,07H,01H,00H,52H  ;A6
DB 18H,08H,01H,00H,52H  ;A7
DB 18H,0DH,01H,00H,52H  ;A8
DB 18H,0EH,01H,00H,52H  ;A9
DB 18H,0FH,01H,00H,52H  ;AA
DB 18H,10H,01H,00H,52H  ;AB
DB 18H,11H,01H,00H,52H  ;AC
DB 18H,12H,01H,00H,52H  ;AD
DB 18H,13H,01H,00H,52H  ;AE
DB 18H,14H,01H,00H,52H  ;AF
DB 05H,0BH,03H,00H,32H  ;B0
DB 01H,01H,00H,00H,12H  ;B1
DB 08H,01H,00H,00H,32H  ;B2
DB 08H,0BH,00H,00H,01H  ;B3
DB 06H,09H,02H,01H,0A3H  ;B4
DB 06H,09H,01H,01H,83H  ;B5
DB 06H,07H,02H,01H,0A3H  ;B6
DB 06H,08H,02H,01H,0A3H  ;B7
DB 06H,0DH,02H,01H,0A3H  ;B8
DB 06H,0EH,02H,01H,0A3H  ;B9
DB 06H,0FH,02H,01H,0A3H  ;BA
DB 06H,10H,02H,01H,0A3H  ;BB
DB 06H,11H,02H,01H,0A3H  ;BC
DB 06H,12H,02H,01H,0A3H  ;BD
DB 06H,13H,02H,01H,0A3H  ;BE
DB 06H,14H,02H,01H,0A3H  ;BF
DB 1FH,01H,00H,00H,52H  ;C0
DB 04H,01H,00H,00H,12H  ;C1
DB 07H,01H,00H,00H,32H  ;C2
DB 07H,0BH,00H,00H,01H  ;C3
DB 29H,09H,00H,00H,01H  ;C4
DB 2AH,09H,01H,00H,52H  ;C5
DB 2AH,09H,07H,00H,01H  ;C6
DB 2AH,09H,08H,00H,01H  ;C7
DB 2AH,09H,0DH,00H,01H  ;C8
DB 2AH,09H,0EH,00H,01H  ;C9
DB 2AH,09H,0FH,00H,01H  ;CA
DB 2AH,09H,10H,00H,01H  ;CB
DB 2AH,09H,11H,00H,01H  ;CC
DB 2AH,09H,12H,00H,01H  ;CD
DB 2AH,09H,13H,00H,01H  ;CE
DB 2AH,09H,14H,00H,01H  ;CF
DB 1EH,01H,00H,00H,52H  ;D0
DB 01H,01H,00H,00H,12H  ;D1
DB 26H,01H,00H,00H,32H  ;D2
DB 26H,0BH,00H,00H,01H  ;D3
DB 09H,09H,00H,00H,01H  ;D4
DB 0CH,01H,01H,00H,83H  ;D5
DB 2BH,09H,07H,00H,01H  ;D6
DB 2BH,09H,08H,00H,01H  ;D7
DB 0CH,0DH,01H,00H,0C2H  ;D8
DB 0CH,0EH,01H,00H,0C2H  ;D9
DB 0CH,0FH,01H,00H,0C2H  ;DA
DB 0CH,10H,01H,00H,0C2H  ;DB
DB 0CH,11H,01H,00H,0C2H  ;DC
DB 0CH,12H,01H,00H,0C2H  ;DD
DB 0CH,13H,01H,00H,0C2H  ;DE
DB 0CH,14H,01H,00H,0C2H  ;DF
DB 1AH,09H,06H,00H,01H  ;E0
DB 04H,01H,00H,00H,12H  ;E1
DB 1AH,09H,07H,00H,01H  ;E2
DB 1AH,09H,08H,00H,01H  ;E3
DB 07H,09H,00H,00H,01H  ;E4
DB 18H,09H,01H,00H,52H  ;E5
DB 18H,09H,07H,00H,01H  ;E6
DB 18H,09H,08H,00H,01H  ;E7
DB 18H,09H,0DH,00H,01H  ;E8
DB 18H,09H,0EH,00H,01H  ;E9
```

```
    DB 18H,09H,0FH,00H,01H  ;EA
    DB 18H,09H,10H,00H,01H  ;EB
    DB 18H,09H,11H,00H,01H  ;EC
    DB 18H,09H,12H,00H,01H  ;ED
    DB 18H,09H,13H,00H,01H  ;EE
    DB 18H,09H,14H,00H,01H  ;EF
    DB 1AH,06H,09H,00H,01H  ;F0
    DB 01H,01H,00H,00H,12H  ;F1
    DB 1AH,07H,09H,00H,01H  ;F2
    DB 1AH,08H,09H,00H,01H  ;F3
    DB 08H,09H,00H,00H,01H  ;F4
    DB 18H,01H,09H,00H,52H  ;F5
    DB 18H,07H,09H,00H,01H  ;F6
    DB 18H,08H,09H,00H,01H  ;F7
    DB 18H,0DH,09H,00H,01H  ;F8
    DB 18H,0EH,09H,00H,01H  ;F9
    DB 18H,0FH,09H,00H,01H  ;FA
    DB 18H,10H,09H,00H,01H  ;FB
    DB 18H,11H,09H,00H,01H  ;FC
    DB 18H,12H,09H,00H,01H  ;FD
    DB 18H,13H,09H,00H,01H  ;FE
    DB 18H,14H,09H,00H,01H  ;FF

;********** DPCOM SUB **********
DPCOM:  PUSH DPL
    CLR C
    MOV A,DPL
    SUBB A,R3
    MOV DPL,A
    MOV A,DPH
    SUBB A,R2
    ORL A,DPL
    POP DPL
    RET
; ********** TEXT SUB **********
TEXT:   MOV R5,#0

TEXT0:  MOV A,@R0       ;BY PASS BLANK
    CJNE A,#' ',TEXT1
    INC R0
    CJNE R0,#INPBUF+20,TEXT0
    RET

TEXT1:  MOV A,@R0       ;READ LOOP
    CJNE A,#' ',S+4
    RET             ;EXIT
    CJNE R0,#INPBUF+20,S+4
    RET
    INC R5
    INC R0
    SJMP TEXT1

; ********** SHIFT4 SUB **********
SHIFT4: PUSH ACC        ;---\
    MOV A,R2
    SWAP A
    ANL A,#0F0H
    MOV R2,A
    MOV A,R3
    SWAP A
    PUSH ACC            ;--\
    ANL A,#0F0H
    MOV R3,A
    POP ACC             ;--/
    ANL A,#0FH
    ORL A,R2
    MOV R2,A
    POP ACC             ;----/
    ANL A,#0FH
    ORL A,R3
    MOV R3,A
    RET

    END
```

19-0094; Rev 7a; 12/96

MAXIM

Microprocessor Supervisory Circuits

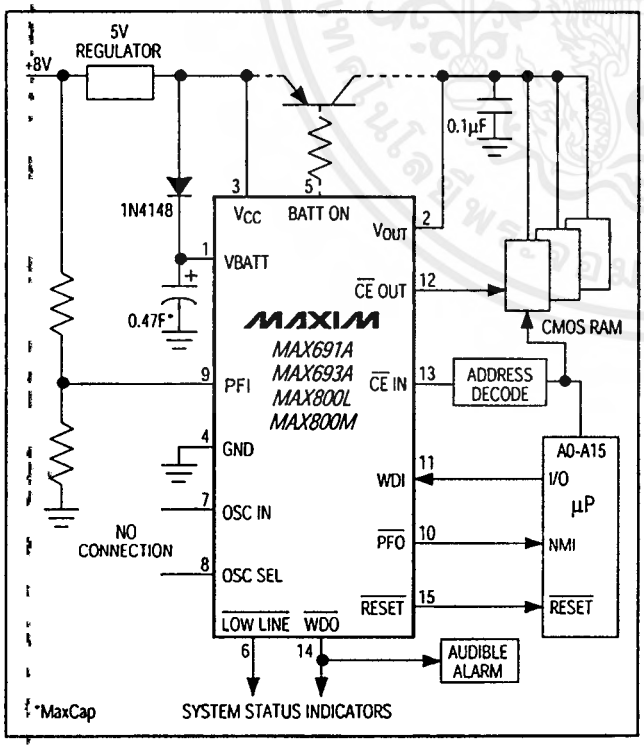
General Description

The MAX691A/MAX693A/MAX800L/MAX800M microprocessor (µP) supervisory circuits are pin-compatible upgrades to the MAX691, MAX693, and MAX695. They improve performance with 30µA supply current, 200ms typ reset active delay on power-up, and 6ns chip-enable propagation delay. Features include write protection of CMOS RAM or EEPROM, separate watchdog outputs, backup-battery switchover, and a $\overline{\text{RESET}}$ output that is valid with V_{CC} down to 1V. The MAX691A/MAX800L have a 4.65V typical reset-threshold voltage, and the MAX693A/MAX800M's reset threshold is 4.4V typical. The MAX800L/MAX800M guarantee power-fail accuracies to ±2%.

Applications

- Computers
- Controllers
- Intelligent Instruments
- Automotive Systems
- Critical µP Power Monitoring

Typical Operating Circuit



Features

- ♦ **200ms Power-OK/Reset Timeout Period**
- ♦ **1µA Standby Current, 30µA Operating Current**
- ♦ **On-Board Gating of Chip-Enable Signals, 10ns Max Delay**
- ♦ **MaxCap™ or SuperCap™ Compatible**
- ♦ **Guaranteed $\overline{\text{RESET}}$ Assertion to V_{CC} = +1V**
- ♦ **Voltage Monitor for Power-Fail or Low-Battery Warning**
- ♦ **Power-Fail Accuracy Guaranteed to ±2% (MAX800L/M)**
- ♦ **Available in 16-Pin Narrow SO and Plastic DIP Packages**

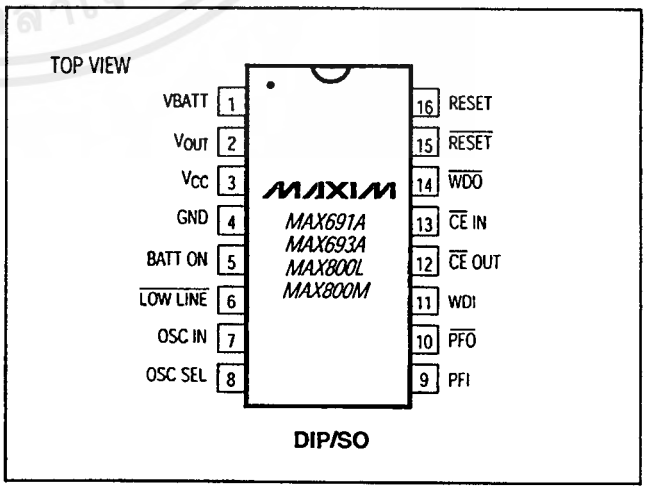
Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX691ACPE	0°C to +70°C	16 Plastic DIP
MAX691ACSE	0°C to +70°C	16 Narrow SO
MAX691ACWE	0°C to +70°C	16 Wide SO
MAX691AC/D	0°C to +70°C	Dice*
MAX691AEPE	-40°C to +85°C	16 Plastic DIP
MAX691AESE	-40°C to +85°C	16 Narrow SO
MAX691AEWE	-40°C to +85°C	16 Wide SO
MAX691AEJE	-40°C to +85°C	16 CERDIP
MAX691AMJE	-55°C to +125°C	16 CERDIP

Ordering Information continued on last page.

** Dice are specified at T_A = +25 °C, DC parameters only.*

Pin Configuration



SuperCap is a registered trademark of Baknor Industries. MaxCap is a registered trademark of The Carborundum Corp.

ABSOLUTE MAXIMUM RATINGS

Terminal Voltage (with respect to GND)
- V_{CC} .....-0.3V to +6V
- VBATT.....-0.3V to +6V
- All Other Inputs .....-0.3V to (V_{OUT} + 0.3V)

Input Current
- V_{CC} Peak.....1.0A
- V_{CC} Continuous.....250mA
- VBATT Peak 250mA
- VBATT Continuous 25mA
- GND, BATT ON 100mA
- All Other Outputs 25mA

Continuous Power Dissipation (T_A = +70°C)
- Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
- Narrow SO (derate 8.70mW/°C above +70°C) 696mW
- Wide SO (derate 9.52mW/°C above +70°C).....762mW
- CERDIP (derate 10.00mW/°C above +70°C).....800mW

Operating Temperature Ranges
- MAX69_AC_ _/MAX800_C_ _ .....0°C to +70°C
- MAX69_AE_ _/MAX800_E_ _.....-40°C to +85°C
- MAX69_AMJE .....-55°C to +125°C

Storage Temperature Range .....-65°C to +160°C

Lead Temperature (soldering, 10sec) .....+300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(MAX691A, MAX800L: V_{CC} = +4.75V to +5.5V, MAX693A, MAX800M: V_{CC} = +4.5V to +5.5V, VBATT = 2.8V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS			MIN	TYP	MAX	UNITS
Operating Voltage Range, V_{CC}, VBATT (Note 1)				0		5.5	V
V_{OUT} Output	V_{CC} = 4.5V	I_{OUT} = 25mA			V_{CC} - 0.02	V_{CC} - 0.05	V
		I_{OUT} = 250mA	MAX69_AC		V_{CC} - 0.2	V_{CC} - 0.3	
			MAX69_AE, MAX800_C/E		V_{CC} - 0.2	V_{CC} - 0.35	
			MAX69_A/M			V_{CC} - 0.40	
		I_{OUT} = 210mA	MAX69_AC/AE, MAX800_C/E		V_{CC} - 0.17	V_{CC} - 0.3V	
V_{CC}-to-V_{OUT} On-Resistance	V_{CC} = 4.5V	MAX69_AC, MAX800_C			0.8	1.2	Ω
		MAX69_AE, MAX800_E			0.8	1.4	
		MAX69_A/M			0.8	1.6	
V_{OUT} in Battery-Backup Mode	VBATT = 4.5V, I_{OUT} = 20mA			VBATT - 0.3			V
	VBATT = 2.8V, I_{OUT} = 10mA			VBATT - 0.25			
	VBATT = 2.0V, I_{OUT} = 5mA			VBATT - 0.15			
VBATT-to-V_{OUT} On-Resistance	VBATT = 4.5V					15	Ω
	VBATT = 2.8V					25	
	VBATT = 2.0V					30	
Supply Current in Normal Operating Mode (Excludes I_{OUT})	V_{CC} > VBATT - 1V				30	100	µA
Supply Current in Battery-Backup Mode (Excludes I_{OUT}) (Note 2)	V_{CC} < VBATT - 1.2V, VBATT = 2.8V	T_A = +25°C			0.04	1	µA
		T_A = T_{MIN} + T_{MIN}				5	
VBATT Standby Current (Note 3)	VBATT + 0.2V ≤ V_{CC}	T_A = +25°C		-0.1		0.02	µA
		T_A = T_{MIN} + T_{MIN}		-1.0		0.02	
Battery Switchover Threshold	Power-up				VBATT + 0.3		V
	Power-down				VBATT - 0.3		

ELECTRICAL CHARACTERISTICS (continued)

(MAX691A, MAX800L: V_{CC} = +4.75V to +5.5V, MAX693A, MAX800M: V_{CC} = +4.5V to +5.5V, VBATT = 2.8V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Battery Switchover Hysteresis			60		mV
BATT ON Output Low Voltage	I_{SINK} = 3.2mA		0.1	0.4	V
	I_{SINK} = 25mA		0.7	1.5	
BATT ON Output Short-Circuit Current	Sink current		60		mA
	Source current	1	15	100	µA
RESET AND WATCHDOG TIMER					
Reset Threshold Voltage	MAX691A, MAX800L	4.50	4.65	4.75	V
	MAX693A, MAX800M	4.25	4.40	4.50	
	MAX800L, T_A = +25°C, V_{CC} falling	4.55		4.70	
	MAX800M, T_A = +25°C, V_{CC} falling	4.30		4.45	
Reset Threshold Hysteresis			15		mV
V_{CC} to RESET Delay	Power-down		80		µs
$\overline{\text{LOW LINE}}$-to-$\overline{\text{RESET}}$ Delay			800		ns
Reset Active Timeout Period, Internal Oscillator	Power-up	140	200	280	ms
Reset Active Timeout Period, External Clock (Note 4)	Power-up		2048		Clock Cycles
Watchdog Timeout Period, Internal Oscillator	Long period	1.0	1.6	2.25	sec
	Short period	70	100	140	ms
Watchdog Timeout Period, External Clock (Note 4)	Long period		4096		Clock Cycles
	Short period		1024		
Minimum Watchdog Input Pulse Width	V_{IL} = 0.8V, V_{IH} = 0.75 x V_{CC}	100			ns
$\overline{\text{RESET}}$ Output Voltage	I_{SINK} = 50µA, V_{CC} = 1V, VBATT = 0V, V_{CC} falling		0.004	0.3	V
	I_{SINK} = 3.2mA, V_{CC} = 4.25V		0.1	0.4	
	I_{SOURCE} = 1.6mA, V_{CC} = 5V	3.5			
$\overline{\text{RESET}}$ Output Short-Circuit Current	Output source current		7	20	mA
$\overline{\text{RESET}}$ Output Voltage Low (Note 5)	I_{SINK} = 3.2mA	0.1	0.4		V
$\overline{\text{LOW LINE}}$ Output Voltage	I_{SINK} = 3.2mA, V_{CC} = 4.25V			0.4	V
	I_{SOURCE} = 1µA, V_{CC} = 5V	3.5			
$\overline{\text{LOW LINE}}$ Output Short-Circuit Current	Output source current	1	15	100	µA
$\overline{\text{WDO}}$ Output Voltage	I_{SINK} = 3.2mA			0.4	V
	I_{SOURCE} = 500µA, V_{CC} = 5V	3.5			
$\overline{\text{WDO}}$ Output Short-Circuit Current	Output source current		3	10	mA
WDI Threshold Voltage (Note 6)	V_{IH}	0.75 x V_{CC}			V
	V_{IL}			0.8	
WDI Input Current	WDI = 0V	-50	-10		µA
	WDI = V_{OUT}		20	50	

ELECTRICAL CHARACTERISTICS (continued)

(MAX691A, MAX800L: V_{CC} = +4.75V to +5.5V, MAX693A, MAX800M: V_{CC} = +4.5V to +5.5V, VBATT = 2.8V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
POWER-FAIL COMPARATOR					
PFI Input Threshold	MAX69_AC/AE/AM, V_{CC} = 5V	1.2	1.25	1.3	V
	MAX800_C/E, V_{CC} = 5V	1.225	1.25	1.275	
PFI Leakage Current			±0.01	±25	nA
$\overline{PFO}$ Output Voltage	I_{SINK} = 3.2mA			0.4	V
	I_{SOURCE} = 1µA, V_{CC} = 5V	3.5			
$\overline{PFO}$ Output Short-Circuit Current	Output source current	1	15	100	µA
PFI-to-PFO Delay	V_{IN} = -20mV, V_{OD} = 15mV		25		µs
	V_{IN} = 20mV, V_{OD} = 15mV		60		
CHIP-ENABLE GATING					
$\overline{CE}$ IN Leakage Current	Disable mode		±0.005	±1	µA
$\overline{CE}$ IN-to-$\overline{CE}$ OUT Resistance (Note 7)	Enable mode		75	150	Ω
$\overline{CE}$ OUT Short-Circuit Current (Reset Active)	Disable mode, $\overline{CE}$ OUT = 0V	0.1	0.75	2.0	mA
$\overline{CE}$ IN-to-$\overline{CE}$ OUT Propagation Delay (Note 8)	50Ω source impedance driver, C_{LOAD} = 50pF		6	10	ns
$\overline{CE}$ OUT Output Voltage High (Reset Active)	V_{CC} = 5V, I_{OUT} = -100µA	3.5			V
	V_{CC} = 0V, VBATT = 2.8V, I_{OUT} = 1µA	2.7			
RESET-to-$\overline{CE}$ OUT Delay	Power-down		12		µs
INTERNAL OSCILLATOR					
OSC IN Leakage Current	OSC SEL = 0V		0.10	±5	µA
OSC IN Input Pull-Up Current	OSC SEL = V_{OUT} or floating, OSC IN = 0V		10	100	µA
OSC SEL Input Pull-Up Current	OSC SEL = 0V		10	100	µA
OSC IN Frequency Range	OSC SEL = 0V		50		kHz
OSC IN External Oscillator Threshold Voltage	V_{IH}	V_{OUT} - 0.3	V_{OUT} - 0.6		V
	V_{IL}		3.65	2.00	
OSC IN Frequency with External Capacitor	OSC SEL = 0V, COSC = 47pF		100		kHz

Note 1: Either V_{CC} or VBATT can go to 0V, if the other is greater than 2.0V.

Note 2: The supply current drawn by the MAX691A/MAX800L/MAX800M from the battery excluding I_{OUT} typically goes to 10µA when (VBATT - 1V) < V_{CC} < VBATT. In most applications, this is a brief period as V_{CC} falls through this region.

Note 3: "+" = battery-discharging current, "--" = battery-charging current.

Note 4: Although presented as typical values, the number of clock cycles for the reset and watchdog timeout periods are fixed and do not vary with process or temperature.

Note 5: RESET is an open-drain output and sinks current only.

Note 6: WDI is internally connected to a voltage divider between V_{OUT} and GND. If unconnected, WDI is driven to 1.6V (typ), disabling the watchdog function.

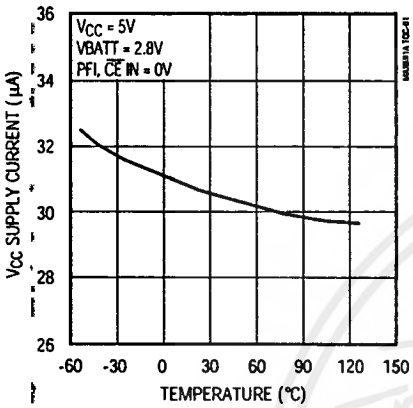
Note 7: The chip-enable resistance is tested with V_{CC} = +4.75V for the MAX691A/MAX800L and V_{CC} = +4.5V for the MAX693A/MAX800M. $\overline{CE}$ IN = $\overline{CE}$ OUT = V_{CC} / 2.

Note 8: The chip-enable propagation delay is measured from the 50% point at $\overline{CE}$ IN to the 50% point at $\overline{CE}$ OUT.

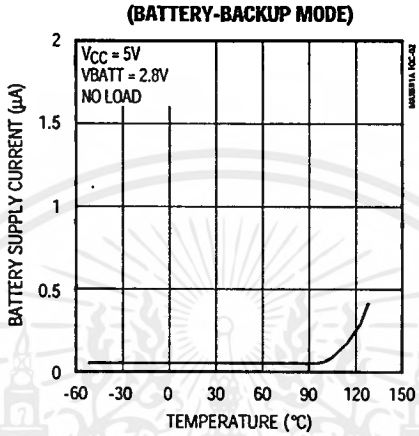
Typical Operating Characteristics

(T_A = +25°C, unless otherwise noted.)

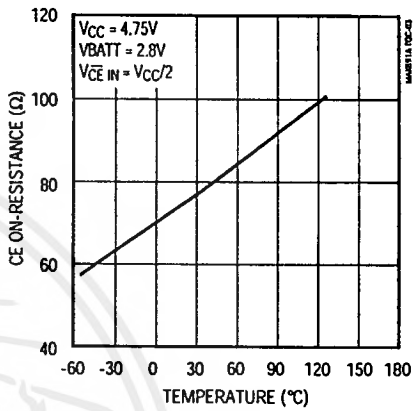
V_{CC} SUPPLY CURRENT vs. TEMPERATURE (NORMAL OPERATING MODE)



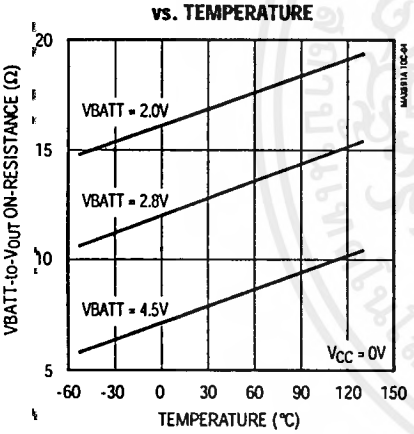
BATTERY SUPPLY CURRENT vs. TEMPERATURE (BATTERY-BACKUP MODE)



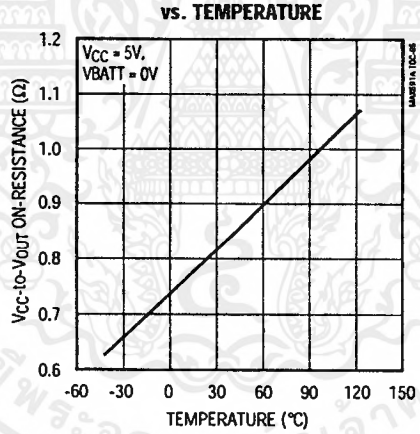
CHIP-ENABLE ON-RESISTANCE vs. TEMPERATURE



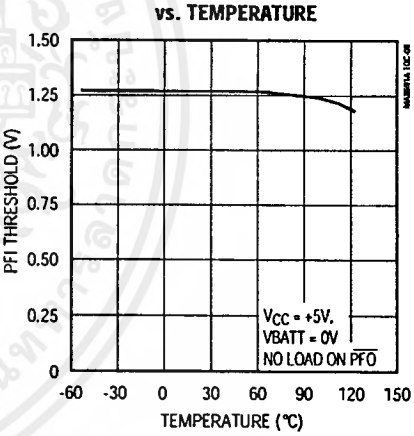
VBATT to V_{OUT} ON-RESISTANCE vs. TEMPERATURE



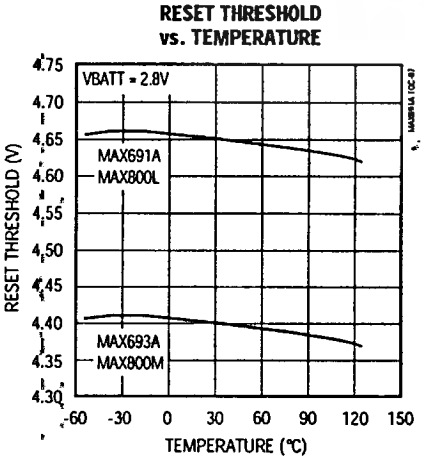
V_{CC} to V_{OUT} ON-RESISTANCE vs. TEMPERATURE



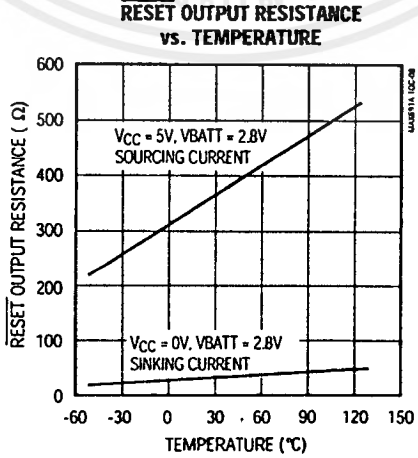
PFI THRESHOLD vs. TEMPERATURE



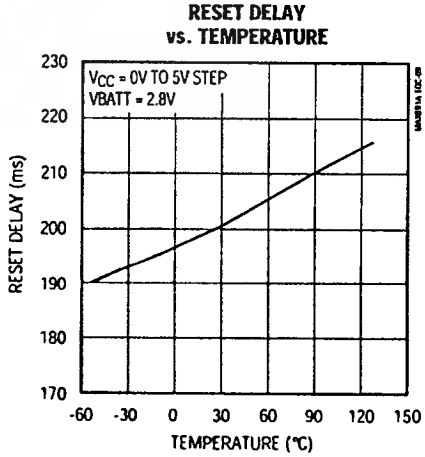
RESET THRESHOLD vs. TEMPERATURE



$\overline{RESET}$ OUTPUT RESISTANCE vs. TEMPERATURE



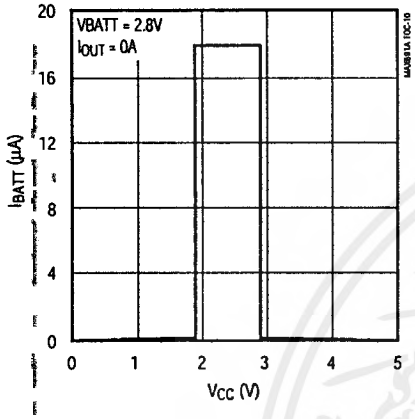
RESET DELAY vs. TEMPERATURE



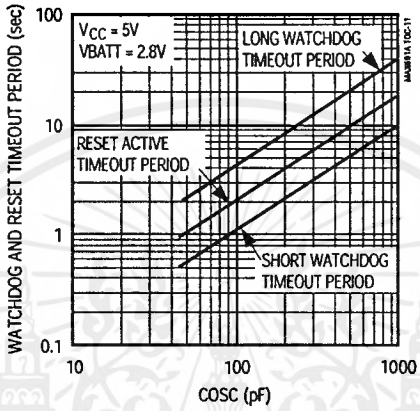
Typical Operating Characteristics (continued)

(T_A = +25°C, unless otherwise noted.)

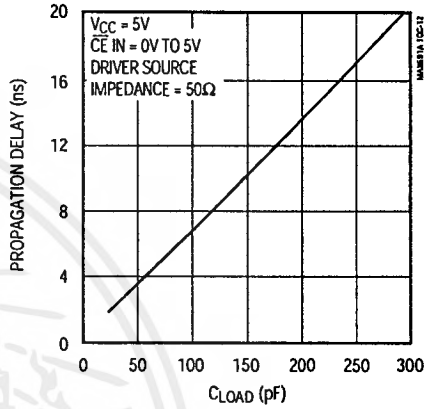
BATTERY CURRENT vs. INPUT SUPPLY VOLTAGE



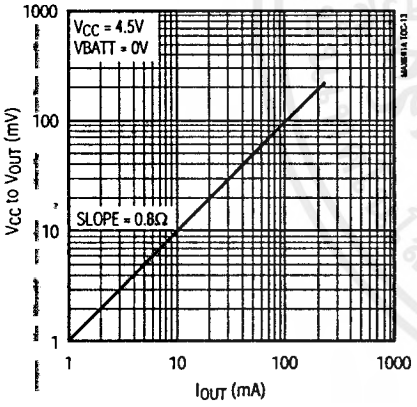
WATCHDOG AND RESET TIMEOUT PERIOD vs. OSC IN TIMING CAPACITOR (COSC)



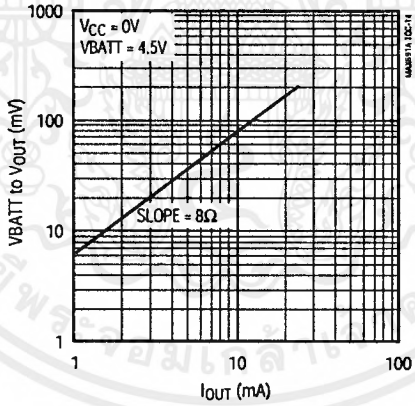
CHIP-ENABLE PROPAGATION DELAY vs. $\overline{CE}$ OUT LOAD CAPACITANCE



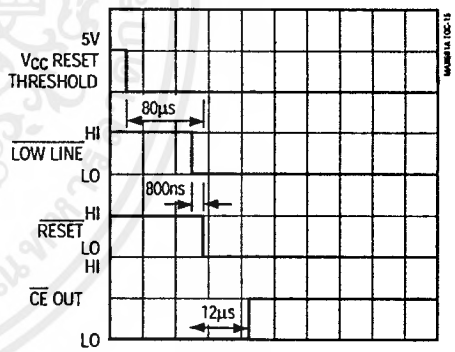
VCC to VOUT vs. OUTPUT CURRENT (NORMAL OPERATING MODE)



VBATT to VOUT vs. OUTPUT CURRENT (BATTERY-BACKUP MODE)



VCC to $\overline{LOW\ LINE}$ AND $\overline{CE}$ OUT DELAY



Pin Description

PIN	NAME	FUNCTION
1	VBATT	Battery-Backup Input. Connect to external battery or capacitor and charging circuit. If backup battery is not used, connect to GND.
2	V_{OUT}	Output Supply Voltage. When V_{CC} is greater than VBATT and above the reset threshold, V_{OUT} connects to V_{CC}. When V_{CC} falls below VBATT and is below the reset threshold, V_{OUT} connects to VBATT. Connect a 0.1µF capacitor from V_{OUT} to GND. Connect V_{OUT} to V_{CC} if no backup battery is used.
3	V_{CC}	Input Supply Voltage, 5V input.
4	GND	Ground. 0V reference for all signals.
5	BATT ON	Battery On Output. When V_{OUT} switches to VBATT, BATT ON goes high. When V_{OUT} switches to V_{CC}, BATT ON goes low. Connect the base of a PNP through a current-limiting resistor to BATT ON for V_{OUT} current requirements greater than 250mA.
6	$\overline{\text{LOW LINE}}$	$\overline{\text{LOW LINE}}$ output goes low when V_{CC} falls below the reset threshold. It returns high as soon as V_{CC} rises above the reset threshold.
7	OSC IN	External Oscillator Input. When OSC SEL is unconnected or driven high, a 10µA pull-up connects from V_{OUT} to OSC IN, the internal oscillator sets the reset and watchdog timeout periods, and OSC IN selects between fast and slow watchdog timeout periods. When OSC SEL is driven low, the reset and watchdog timeout periods may be set either by a capacitor from OSC IN to ground or by an external clock at OSC IN (Figure 3).
8	OSC SEL	Oscillator Select. When OSC SEL is unconnected or driven high, the internal oscillator sets the reset delay and watchdog timeout period. When OSC SEL is low, the external oscillator input (OSC IN) is enabled (Table 1). OSC SEL has a 10µA internal pull-up.
9	PFI	Power-Fail Input. This is the noninverting input to the power-fail comparator. When PFI is less than 1.25V, $\overline{\text{PFO}}$ goes low. When PFI is not used, connect PFI to GND or V_{OUT}.
10	$\overline{\text{PFO}}$	Power-Fail Output. This is the output of the power-fail comparator. $\overline{\text{PFO}}$ goes low when PFI is less than 1.25V. This is an uncommitted comparator, and has no effect on any other internal circuitry.
11	WDI	Watchdog Input. WDI is a three-level input. If WDI remains either high or low for longer than the watchdog timeout period, $\overline{\text{WDO}}$ goes low and reset is asserted for the reset timeout period. $\overline{\text{WDO}}$ remains low until the next transition at WDI. Leaving WDI unconnected disables the watchdog function. WDI connects to an internal voltage divider between V_{OUT} and GND, which sets it to mid-supply when left unconnected.
12	$\overline{\text{CE}}$ OUT	Chip-Enable Output. $\overline{\text{CE}}$ OUT goes low only when $\overline{\text{CE}}$ IN is low and V_{CC} is above the reset threshold. If CE IN is low when reset is asserted, $\overline{\text{CE}}$ OUT will stay low for 15µs or until $\overline{\text{CE}}$ IN goes high, whichever occurs first.
13	$\overline{\text{CE}}$ IN	Chip-Enable Input. The input to chip-enable gating circuit. If $\overline{\text{CE}}$ IN is not used, connect $\overline{\text{CE}}$ IN to GND or V_{OUT}.
14	$\overline{\text{WDO}}$	Watchdog Output. If WDI remains high or low longer than the watchdog timeout period, $\overline{\text{WDO}}$ goes low and reset is asserted for the reset timeout period. $\overline{\text{WDO}}$ returns high on the next transition at WDI. $\overline{\text{WDO}}$ remains high if WDI is unconnected.
15	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$ Output goes low whenever V_{CC} falls below the reset threshold. $\overline{\text{RESET}}$ will remain low typically for 200ms after V_{CC} crosses the reset threshold on power-up.
16	RESET	RESET is an active-high output. It is open drain, and the inverse of $\overline{\text{RESET}}$.

Detailed Description

$\overline{\text{RESET}}$ and RESET Outputs

The MAX691A/MAX693A/MAX800L/MAX800M's $\overline{\text{RESET}}$ and RESET outputs ensure that the µP (with reset inputs asserted either high or low) powers up in a known state, and prevents code-execution errors during power-down or brownout conditions.

The $\overline{\text{RESET}}$ output is active low, and typically sinks 3.2mA at 0.1V saturation voltage in its active state. When deasserted, $\overline{\text{RESET}}$ sources 1.6mA at typically V_{OUT} - 0.5V. RESET output is open drain, active high, and typically sinks 3.2mA with a saturation voltage of 0.1V. When no backup battery is used, $\overline{\text{RESET}}$ output is guaranteed to be valid down to V_{CC} = 1V, and an external 10kΩ pull-down resistor on $\overline{\text{RESET}}$ insures that it will be valid with V_{CC} down to GND (Figure 1). As V_{CC} goes below 1V, the gate drive to the $\overline{\text{RESET}}$ output switch reduces accordingly, increasing the $R_{DS(ON)}$ and the saturation voltage. The 10kΩ pull-down resistor insures the parallel combination of switch plus resistor is around 10kΩ and the output saturation voltage is below 0.4V while sinking 40µA. When using a 10kΩ external pull-down resistor, the high state for $\overline{\text{RESET}}$ output with V_{CC} = 4.75V will be 4.5V typical. For battery voltages ≥ 2V connected to VBATT, RESET and $\overline{\text{RESET}}$ remain valid for V_{CC} from 0V to 5.5V.

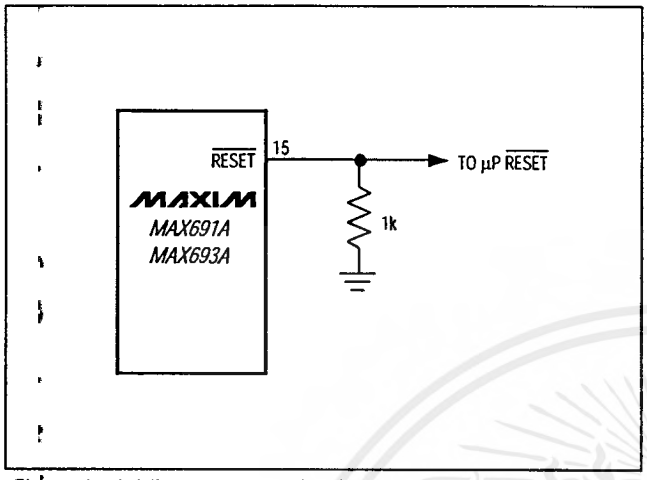


Figure 1. Adding an external pull-down resistor ensures RESET is valid with V_{CC} down to GND.

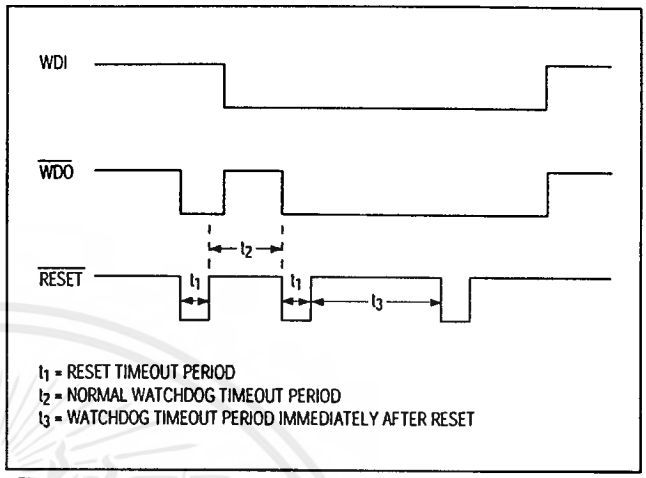


Figure 2. Watchdog Timeout Period and Reset Active Time

RESET and $\overline{\text{RESET}}$ are asserted when V_{CC} falls below the reset threshold (4.65V for the MAX691A/MAX800L, 4.4V for the MAX693A/MAX800M) and remain asserted for 200ms typ after V_{CC} rises above the reset threshold on power-up (Figure 5). The devices' battery-switchover comparator does not affect reset assertion. However, both reset outputs are asserted in battery-backup mode since V_{CC} must be below the reset threshold to enter this mode.

Watchdog Function

The watchdog monitors µP activity via the Watchdog Input (WDI). If the µP becomes inactive, $\overline{\text{RESET}}$ and RESET are asserted. To use the watchdog function, connect WDI to a bus line or µP I/O line. If WDI remains high or low for longer than the watchdog time-out period (1.6sec nominal), $\overline{\text{WDO}}$, RESET, and $\overline{\text{RESET}}$ are asserted (see *RESET and $\overline{\text{RESET}}$ Outputs* section, and the *Watchdog Output* discussion on this page).

Watchdog Input

A change of state (high to low, low to high, or a minimum 100ns pulse) at the WDI during the watchdog period resets the watchdog timer. The watchdog default timeout is 1.6sec.

To disable the watchdog function, leave WDI floating. An internal resistor network (100kΩ equivalent impedance at WDI) biases WDI to approximately 1.6V. Internal comparators detect this level and disable the watchdog timer. When V_{CC} is below the reset threshold, the watchdog function is disabled and WDI is disconnected from its internal resistor network, thus becoming high impedance.

Watchdog Output

The Watchdog Output ($\overline{\text{WDO}}$) remains high if there is a transition or pulse at WDI during the watchdog timeout period. The watchdog function is disabled and $\overline{\text{WDO}}$ is a logic high when V_{CC} is below the reset threshold, battery-backup mode is enabled, or WDI is an open circuit. In watchdog mode, if no transition occurs at WDI during the watchdog timeout period, RESET and $\overline{\text{RESET}}$ are asserted for the reset timeout period (200ms typical). $\overline{\text{WDO}}$ goes low and remains low until the next transition at WDI (Figure 2). If WDI is held high or low indefinitely, RESET and $\overline{\text{RESET}}$ will generate 200ms pulses every 1.6sec. $\overline{\text{WDO}}$ has a 2 x TTL output characteristic.

Selecting an Alternative Watchdog and Reset Timeout Period

The OSC SEL and OSC IN inputs control the watchdog and reset timeout periods. Floating OSC SEL and OSC IN or tying them both to V_{OUT} selects the nominal 1.6sec watchdog timeout period and 200ms reset timeout period. Connecting OSC IN to GND and floating or connecting OSC SEL to V_{OUT} selects the 100ms normal watchdog timeout delay and 1.6sec delay immediately after reset. The reset timeout delay remains 200ms (Figure 2). Select alternative timeout periods by connecting OSC SEL to GND and connecting a capacitor between OSC IN and GND, or by externally driving OSC IN (Table 1 and Figure 3). OSC IN is internally connected to a ±100nA (typ) current source that charges and discharges the timing capacitor to create the oscillator frequency, which sets the reset and watchdog timeout periods (see *Connecting a Timing Capacitor at OSC IN* in the *Applications Information* section).

Table 1. Reset Pulse Width and Watchdog Timeout Selections

OSC SEL	OSC IN	Watchdog Timeout Period		Reset Timeout Period
		Normal	Immediately After Reset	
Low	External Clock Input	1024 clks	4096 clks	2048 clks
Low	External Capacitor	(600/47pF x C)ms	(2.4/47pF x C)sec	(1200/47pF x C)ms
Floating	Low	100ms	1.6sec	200ms
Floating	Floating	1.6sec	1.6sec	200ms

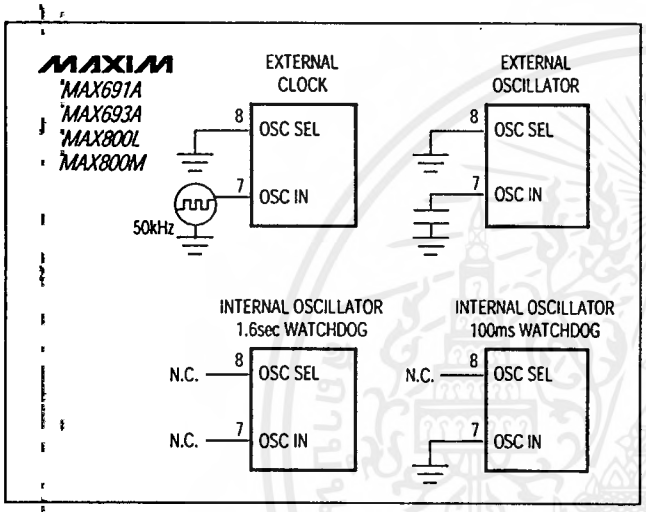


Figure 3. Oscillator Circuits

Chip-Enable Signal Gating

The MAX691A/MAX693A/MAX800L/MAX800M provide internal gating of chip-enable (CE) signals to prevent erroneous data from being written to CMOS RAM in the event of a power failure. During normal operation, the CE gate is enabled and passes all CE transitions. When reset is asserted, this path becomes disabled, preventing erroneous data from corrupting the CMOS RAM. All these parts use a series transmission gate from $\overline{CE}$ IN to $\overline{CE}$ OUT (Figure 4).

The 10ns max CE propagation delay from $\overline{CE}$ IN to $\overline{CE}$ OUT enables the parts to be used with most µPs.

Chip-Enable Input

The Chip-Enable Input ($\overline{CE}$ IN) is high impedance (disabled mode) while RESET and $\overline{RESET}$ are asserted.

During a power-down sequence where V_{CC} falls below the reset threshold or a watchdog fault, $\overline{CE}$ IN assumes a high-impedance state when the voltage at $\overline{CE}$ IN goes high or 15µs after reset is asserted, whichever occurs first (Figure 5).

During a power-up sequence, $\overline{CE}$ IN remains high impedance, regardless of $\overline{CE}$ IN activity, until reset is deasserted following the reset timeout period.

In the high-impedance mode, the leakage currents into this terminal are ±1µA max over temperature. In the low-impedance mode, the impedance of $\overline{CE}$ IN appears as a 75Ω resistor in series with the load at $\overline{CE}$ OUT.

The propagation delay through the CE transmission gate depends on both the source impedance of the drive to $\overline{CE}$ IN and the capacitive loading on the Chip-Enable Output ($\overline{CE}$ OUT) (see Chip-Enable Propagation Delay vs. $\overline{CE}$ OUT Load Capacitance in the *Typical Operating Characteristics*). The CE propagation delay is production tested from the 50% point of $\overline{CE}$ IN to the 50% point of $\overline{CE}$ OUT using a 50Ω driver and 50pF of load capacitance (Figure 6). For minimum propagation delay, minimize the capacitive load at $\overline{CE}$ OUT, and use a low output-impedance driver.

Chip-Enable Output

In the enabled mode, the impedance of $\overline{CE}$ OUT is equivalent to 75Ω in series with the source driving $\overline{CE}$ IN. In the disabled mode, the 75Ω transmission gate is off and $\overline{CE}$ OUT is actively pulled to V_{OUT}. This source turns off when the transmission gate is enabled.

$\overline{LOW\ LINE}$ Output

$\overline{LOW\ LINE}$ is the buffered output of the reset threshold comparator. $\overline{LOW\ LINE}$ typically sinks 3.2mA at 0.1V. For normal operation (V_{CC} above the $\overline{LOW\ LINE}$ threshold), $\overline{LOW\ LINE}$ is pulled to V_{OUT}.

Power-Fail Comparator

The power-fail comparator is an uncommitted comparator that has no effect on the other functions of the IC. Common uses include low-battery indication (Figure 7), and early power-fail warning (see *Typical Operating Circuit*).

Power-Fail Input

Power Fail Input (PFI) is the input to the power-fail comparator. It has a guaranteed input leakage of ±25nA max over temperature. The typical comparator delay is 25µs from V_{IL} to V_{OL} (power failing), and 60µs from V_{IH} to V_{OH} (power being restored). If PFI is not used, connect it to ground.

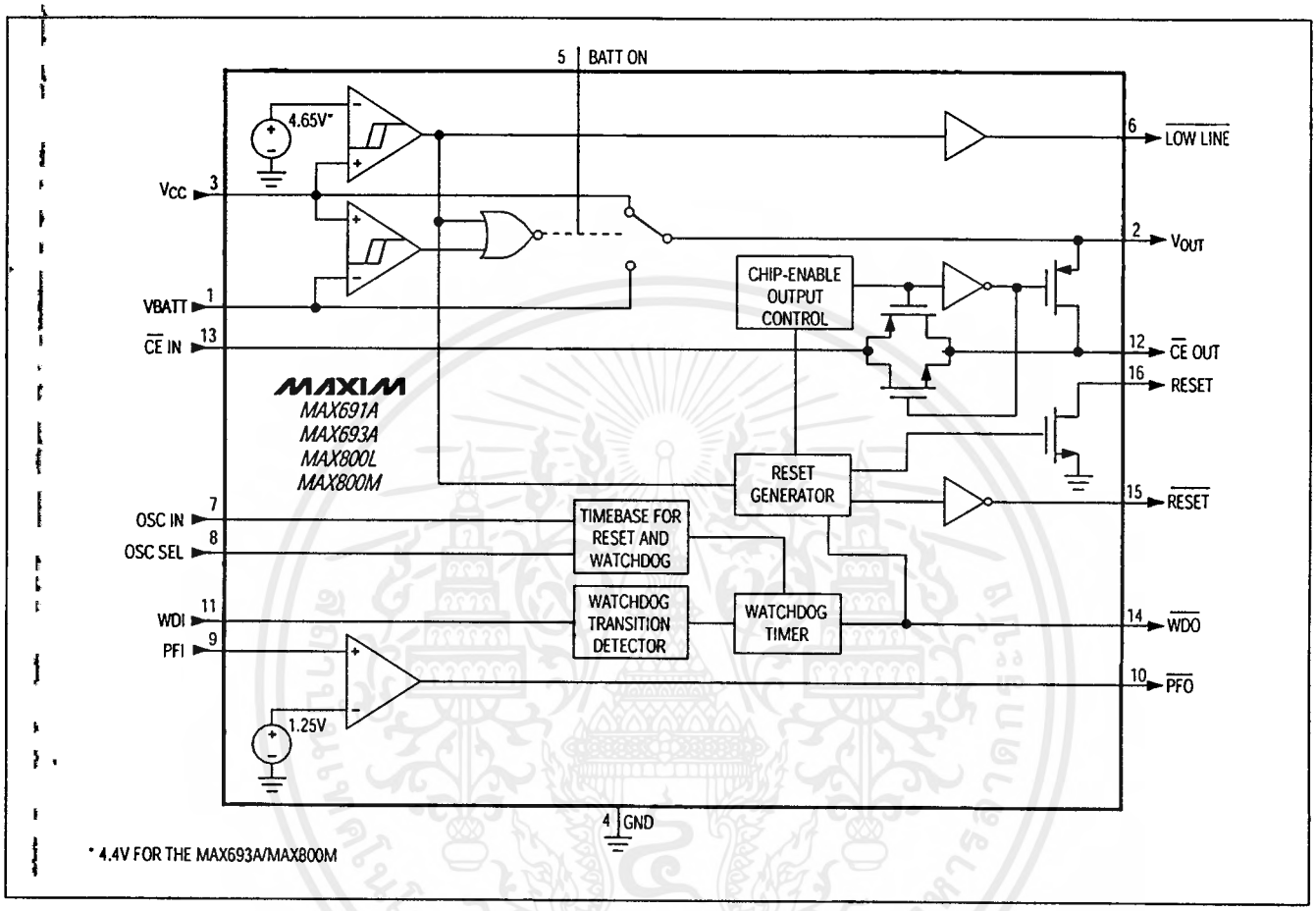


Figure 4. MAX691A/MAX693A/MAX800L/MAX800M Block Diagram

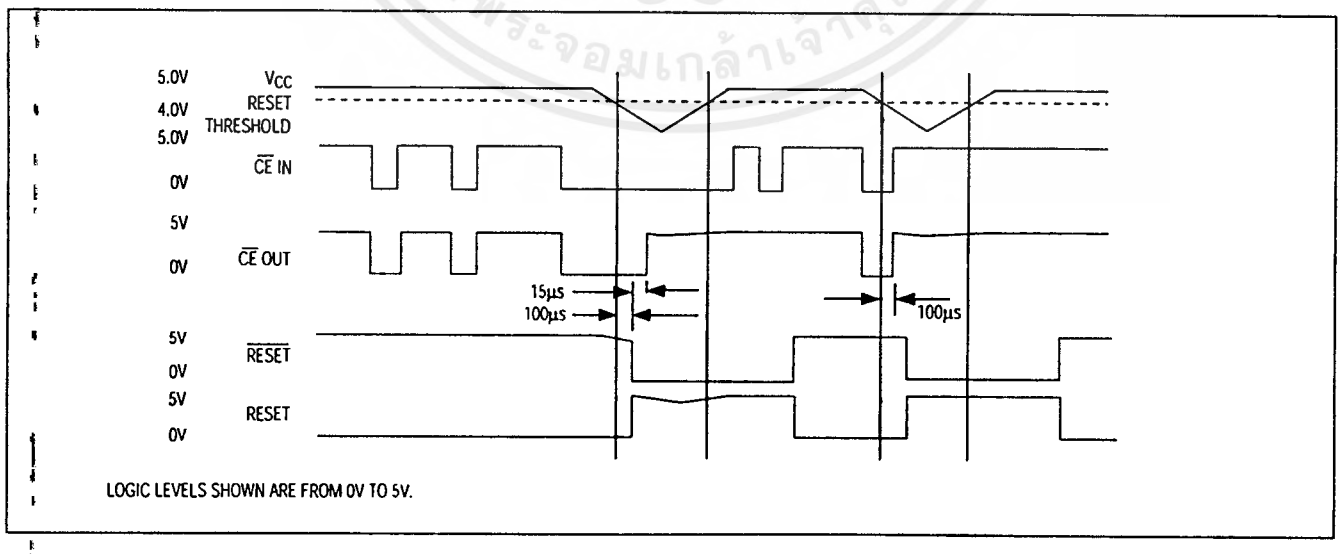


Figure 5. Reset and Chip-Enable Timing

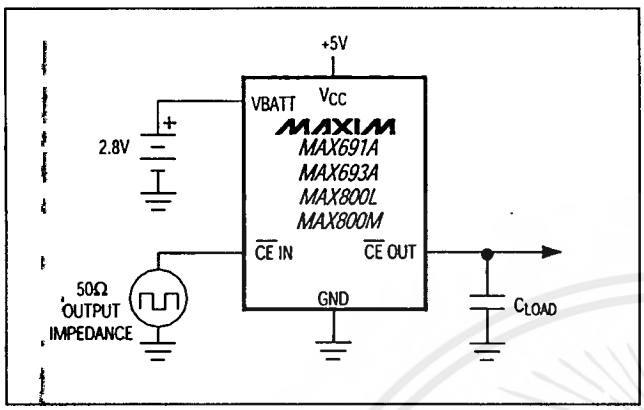


Figure 6. CE Propagation Delay Test Circuit

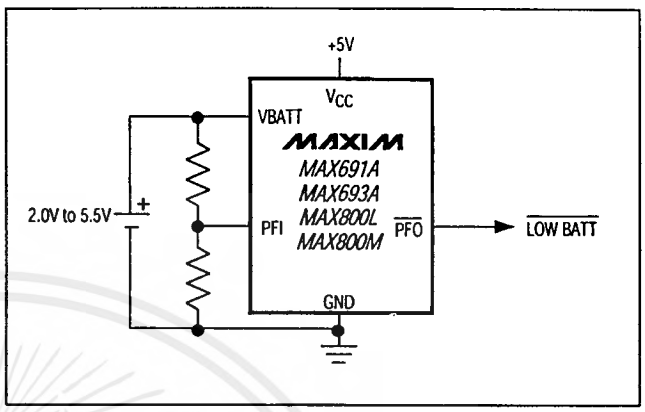


Figure 7. Low-Battery Indicator

Table 2. Input and Output Status in Battery-Backup Mode

PIN	NAME	STATUS
1	VBATT	Supply current is 1µA max.
2	VOUT	VOUT is connected to VBATT through an internal PMOS switch.
3	VCC	Battery switchover comparator monitors VCC for active switchover.
4	GND	GND 0V, 0V reference for all signals.
5	BATT ON	Logic high. The open-circuit output is equal to VOUT.
6	$\overline{\text{LOWLINE}}$	Logic low*
7	OSC IN	OSC IN is ignored.
8	OSC SEL	OSC SEL is ignored.
9	PFI	The power-fail comparator remains active in the battery-backup mode for VCC ≥ VBATT - 1.2V typ.
10	$\overline{\text{PFO}}$	The power-fail comparator remains active in the battery-backup mode for VCC ≥ VBATT - 1.2V typ. Below this voltage, PFO is forced low.
11	WDI	Watchdog is ignored.
12	$\overline{\text{CE}}$ OUT	Logic high. The open-circuit voltage is equal to VOUT.
13	$\overline{\text{CE}}$ IN	High impedance
14	$\overline{\text{WDO}}$	Logic high. The open-circuit voltage is equal to VOUT.
15	$\overline{\text{RESET}}$	Logic low*
16	RESET	High impedance*

** V_{CC} must be below the reset threshold to enter battery-backup mode.*

Power-Fail Output

The Power-Fail Output ($\overline{\text{PFO}}$) goes low when PFI goes below 1.25V. It typically sinks 3.2mA with a saturation voltage of 0.1V. With PFI above 1.25V, $\overline{\text{PFO}}$ is actively pulled to V_{OUT}.

Battery-Backup Mode

Two conditions are required to switch to battery-backup mode: 1) V_{CC} must be below the reset threshold, and 2) V_{CC} must be below VBATT. Table 2 lists the status of the inputs and outputs in battery-backup mode.

Battery On Output

The Battery On (BATT ON) output indicates the status of the internal V_{CC}/battery-switchover comparator, which controls the internal V_{CC} and VBATT switches. For V_{CC} greater than VBATT (ignoring the small hysteresis effect), BATT ON typically sinks 3.2mA at 0.1V saturation voltage. In battery-backup mode, this terminal sources approximately 10µA from V_{OUT}. Use BATT ON to indicate battery-switchover status or to supply base drive to an external pass transistor for higher-current applications (see *Typical Operating Circuit*).

Input Supply Voltage

The Input Supply Voltage (V_{CC}) should be a regulated 5V. V_{CC} connects to V_{OUT} via a parallel diode and a large PMOS switch. The switch carries the entire current load for currents less than 250mA. The parallel diode carries any current in excess of 250mA. Both the switch and the diode have impedances less than 1Ω each. The maximum continuous current is 250mA, but power-on transients may reach a maximum of 1A.

Battery-Backup Input

The Battery-Backup Input (VBATT) is similar to the V_{CC} input except the PMOS switch and parallel diode are much smaller. Accordingly, the on-resistances of the diode and the switch are each approximately 10Ω. Continuous current should be limited to 25mA and peak currents (only during power-up) limited to 250mA. The reverse leakage of this input is less than 1µA over temperature and supply voltage (Figure 8).

Output Supply Voltage

The Output Supply Voltage (V_{OUT}) pin is internally connected to the substrate of the IC and supplies current to the external system and internal circuitry. All open-circuit outputs will, for example, assume the V_{OUT} voltage in their high states rather than the V_{CC} voltage. At the maximum source current of 250mA, V_{OUT} will typically be 200mV below V_{CC}. Decouple this terminal with a 0.1µF capacitor.

Applications Information

The MAX691A/MAX693A/MAX800L/MAX800M are not short-circuit protected. Shorting V_{OUT} to ground, other than power-up transients such as charging a decoupling capacitor, destroys the device.

All open-circuit outputs swing between V_{OUT} and GND rather than V_{CC} and GND.

If long leads connect to the chip inputs, insure that these leads are free from ringing and other conditions that would forward bias the chip's protection diodes.

There are three distinct modes of operation:

1) Normal operating mode with all circuitry powered up. Typical supply current from V_{CC} is 35µA while only leakage currents flow from the battery.
2) Battery-backup mode where V_{CC} is typically within 0.7V below VBATT. All circuitry is powered up and the supply current from the battery is typically less than 60µA.
3) Battery-backup mode where V_{CC} is less than VBATT by at least 0.7V. VBATT supply current is 1µA max.

Using SuperCap™ or MaxCap™ with the MAX691A/MAX693A/MAX800L/MAX800M

VBATT has the same operating voltage range as V_{CC}, and the battery switchover threshold voltages are typically ±30mV centered at VBATT, allowing use of a SuperCap and a simple charging circuit as a backup source (Figure 9).

If V_{CC} is above the reset threshold and VBATT is 0.5V above V_{CC}, current flows to V_{OUT} and V_{CC} from VBATT until the voltage at VBATT is less than 0.5V above V_{CC}. For example, with a SuperCap connected to VBATT and through a diode to V_{CC}, if V_{CC} quickly changes from 5.4V to 4.9V, the capacitor discharges through V_{OUT} and V_{CC} until VBATT reaches 5.1V typ. Leakage current through the SuperCap charging diode and the internal power diode eventually discharges the SuperCap to V_{CC}. Also, if V_{CC} and VBATT start from 0.1V above the reset threshold and power is lost at V_{CC}, the SuperCap on VBATT discharges through V_{CC} until VBATT reaches the reset threshold; then the battery-backup mode is initiated and the current through V_{CC} goes to zero.

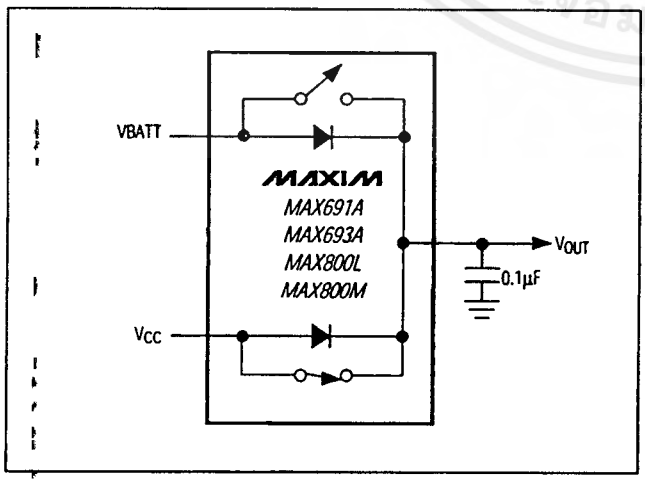


Figure 8. V_{CC} and VBATT to V_{OUT} Switch

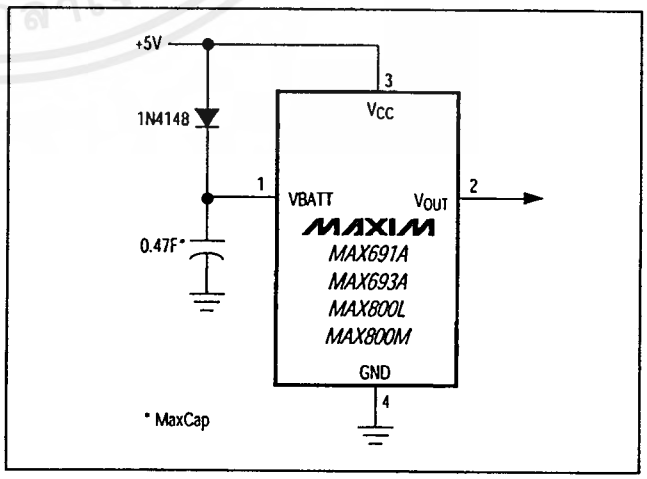


Figure 9. SuperCap or MaxCap on VBATT

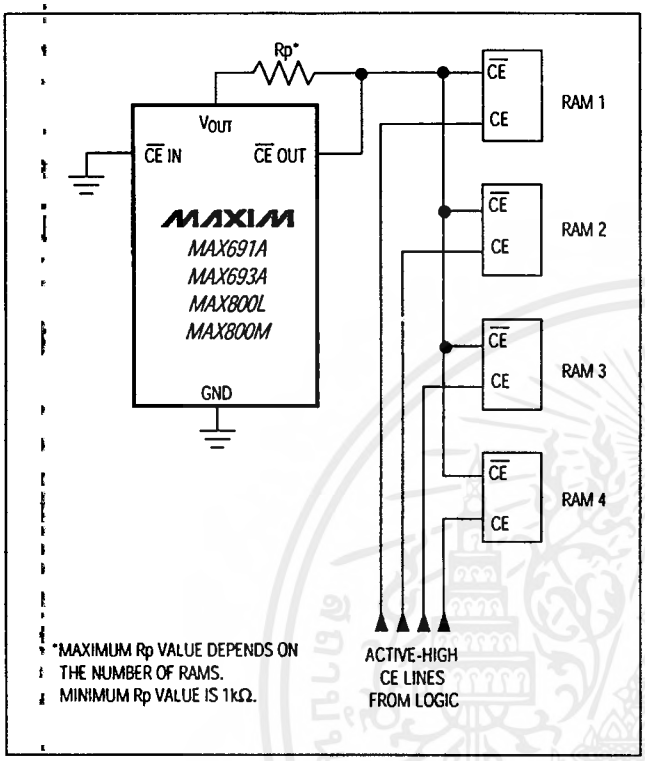


Figure 10. Alternate CE Gating

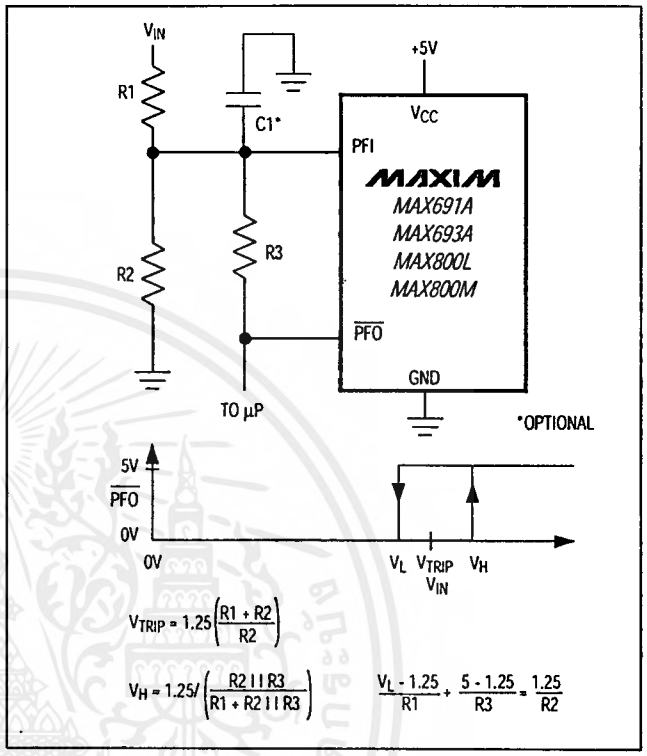


Figure 11. Adding Hysteresis to the Power-Fail Comparator

Using Separate Power Supplies for VBATT and V_{CC}

If using separate power supplies for V_{CC} and VBATT, VBATT must be less than 0.3V above V_{CC} when V_{CC} is above the reset threshold. As described in the previous section, if VBATT exceeds this limit and power is lost at V_{CC}, current flows continuously from VBATT to V_{CC} via the VBATT-to-V_{OUT} diode and the V_{OUT}-to-V_{CC} switch until the circuit is broken (Figure 8).

Alternate Chip-Enable Gating

Using memory devices with both CE and $\overline{CE}$ inputs allows the CE loop to be bypassed. To do this, connect $\overline{CE}$ IN to ground, pull up $\overline{CE}$ OUT to V_{OUT}, and connect $\overline{CE}$ OUT to the $\overline{CE}$ input of each memory device (Figure 10). The CE input of each part then connects directly to the chip-select logic, which does not have to be gated.

Adding Hysteresis to the Power-Fail Comparator

Hysteresis adds a noise margin to the power-fail comparator and prevents repeated triggering of $\overline{PFO}$ when V_{IN} is near the power-fail comparator trip point. Figure 11 shows how to add hysteresis to the power-fail com-

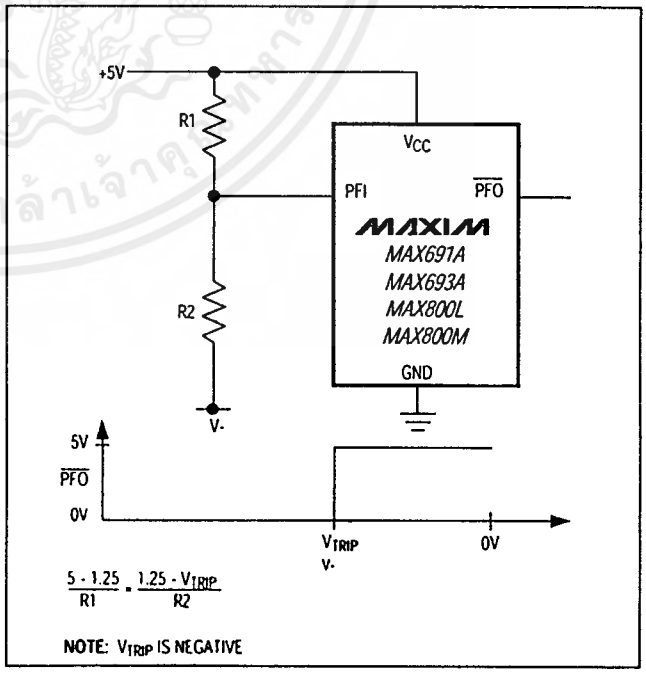


Figure 12. Monitoring a Negative Voltage

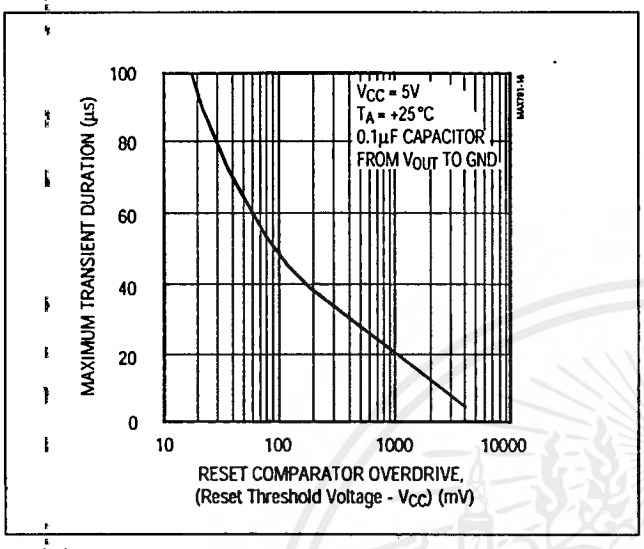


Figure 13. Maximum Transient Duration without Causing a Reset Pulse vs. Reset Comparator Overdrive

parator. Select the ratio of R1 and R2 such that PFI sees 1.25V when V_{IN} falls to the desired trip point (V_{TRIP}). Resistor R3 adds hysteresis. It will typically be an order of magnitude greater than R1 or R2. The current through R1 and R2 should be at least 1µA to ensure that the 25nA (max) PFI input current does not shift the trip point. R3 should be larger than 10kΩ to prevent it from loading down the $\overline{PFO}$ pin. Capacitor C1 adds noise rejection.

Monitoring a Negative Voltage

The power-fail comparator can be used to monitor a negative supply voltage using Figure 12's circuit. When the negative supply is valid, $\overline{PFO}$ is low. When the negative supply voltage drops, $\overline{PFO}$ goes high. This circuit's accuracy is affected by the PFI threshold tolerance, the V_{CC} voltage, and resistors R1 and R2.

Backup-Battery Replacement

The backup battery may be disconnected while V_{CC} is above the reset threshold. No precautions are necessary to avoid spurious reset pulses.

Negative-Going V_{CC} Transients

While issuing resets to the µP during power-up, power-down, and brownout conditions, these supervisors are relatively immune to short-duration, negative-going V_{CC} transients (glitches). It is usually undesirable to reset the µP when V_{CC} experiences only small glitches.

Figure 13 shows maximum transient duration vs. reset-comparator overdrive, for which reset pulses are **not** generated. The graph was produced using negative-going V_{CC} pulses, starting at 5V and ending below the reset threshold by the magnitude indicated (reset comparator overdrive). The graph shows the maximum pulse width a negative-going V_{CC} transient may typically have without causing a reset pulse to be issued. As the amplitude of the transient increases (i.e., goes farther below the reset threshold), the maximum allowable pulse width decreases. Typically, a V_{CC} transient that goes 100mV below the reset threshold and lasts for 40µs or less will not cause a reset pulse to be issued.

A 100nF bypass capacitor mounted close to the V_{CC} pin provides additional transient immunity.

Connecting a Timing Capacitor at OSC IN

When OSC SEL is connected to ground, OSC IN disconnects from its internal 10µA (typ) pull-up and is internally connected to a ±100nA current source. When a capacitor is connected from OSC IN to ground (to select alternative reset and watchdog timeout periods), the current source charges and discharges the timing capacitor to create the oscillator that controls the reset and watchdog timeout period. To prevent timing errors or oscillator start-up problems, minimize external current leakage sources at this pin, and locate the capacitor as close to OSC IN as possible. The sum of PC-board leakage plus OSC capacitor leakage must be small compared to ±100nA.

Maximum V_{CC} Fall Time

The V_{CC} fall time is limited by the propagation delay of the battery switchover comparator and should not exceed 0.03V/µs. A standard rule of thumb for filter capacitance on most regulators is on the order of 100µF per amp of current. When the power supply is shut off or the main battery is disconnected, the associated initial V_{CC} fall rate is just the inverse or 1A/100µF = 0.01V/µs. The V_{CC} fall rate decreases with time as V_{CC} falls exponentially, which more than satisfies the maximum fall-time requirement.

Watchdog Software Considerations

A way to help the watchdog timer keep a closer watch on software execution involves setting and resetting the watchdog input at different points in the program, rather than "pulsing" the watchdog input high-low-high or low-high-low. This technique avoids a "stuck" loop where the watchdog timer continues to be reset within the loop, keeping the watchdog from timing out. Figure 14 shows an example flow diagram where the I/O driving the watchdog input is set high at the beginning of the program, set low at the beginning of every subroutine or loop, then set high again when the program returns to the beginning. If the program should "hang" in any subroutine, the I/O is continually set low and the watchdog timer is allowed to time out, causing a reset or interrupt to be issued.

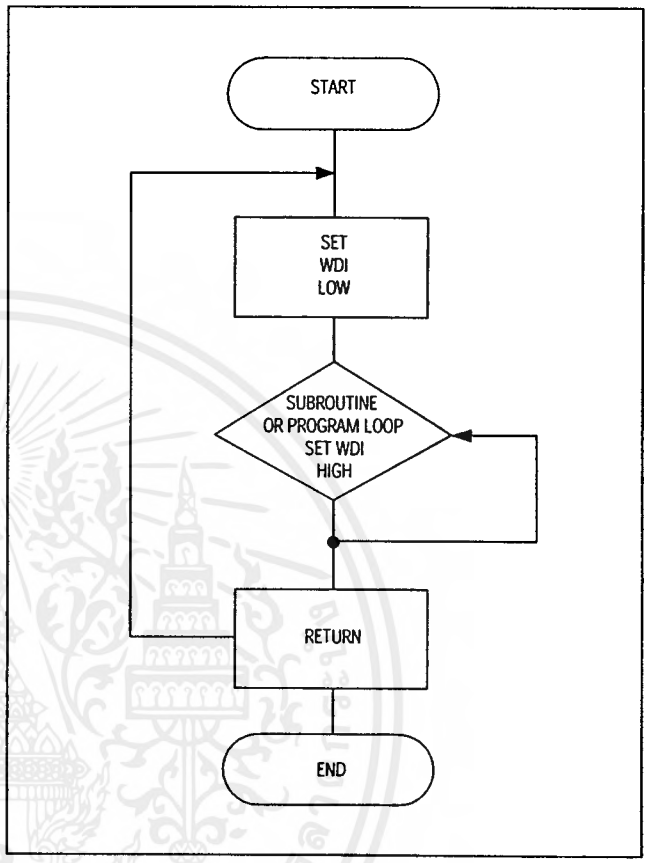


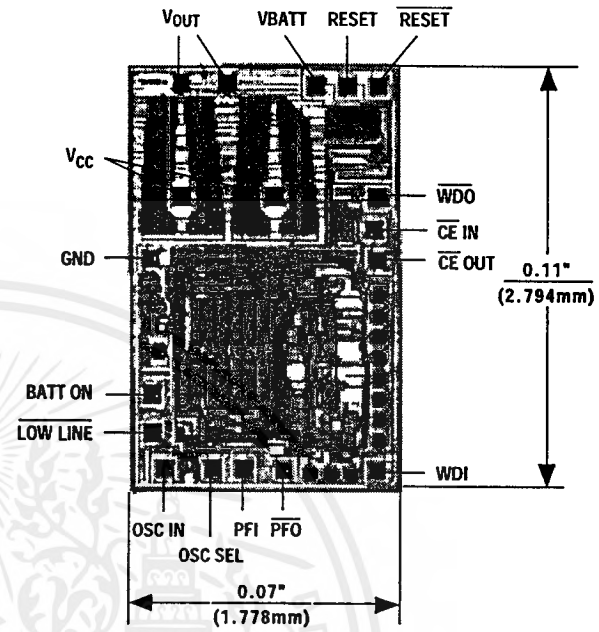
Figure 14. Watchdog Flow Diagram

Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE
MAX693ACPE	0°C to +70°C	16 Plastic DIP
MAX693ACSE	0°C to +70°C	16 Narrow SO
MAX693ACWE	0°C to +70°C	16 Wide SO
MAX693AC/D	0°C to +70°C	Dice*
MAX693AEPE	-40°C to +85°C	16 Plastic SO
MAX693AESE	-40°C to +85°C	16 Narrow SO
MAX693AEWE	-40°C to +85°C	16 Wide SO
MAX693AEJE	-40°C to +85°C	16 CERDIP
MAX693AMJE	-55°C to +125°C	16 CERDIP
MAX800LCPE	0°C to +70°C	16 Plastic DIP
MAX800LCSE	0°C to +70°C	16 Narrow SO
MAX800LEPE	-40°C to +85°C	16 Plastic DIP
MAX800LESE	-40°C to +85°C	16 Narrow SO
MAX800MCPE	0°C to +70°C	16 Plastic DIP
MAX800MCSE	0°C to +70°C	16 Narrow SO
MAX800MEPE	-40°C to +85°C	16 Plastic DIP
MAX800MESE	-40°C to +85°C	16 Narrow SO

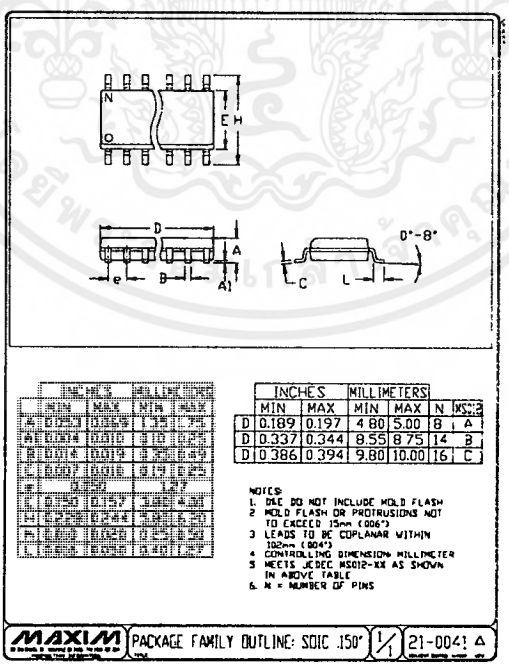
** Dice are specified at T_A = +25°C, DC parameters only.*

Chip Topography



TRANSISTOR COUNT: 729
SUBSTRATE CONNECTED TO VOUT

Package Information



กิตติกรรมประกาศ

ขอขอบคุณ ผศ.ดร.สุรพันธุ์ เอื้อไพบูลย์ ที่ให้คำแนะนำแนวทางการจัดทำโครงงาน ตลอดจนให้ความรู้ ความเข้าใจในการปฏิบัติงาน

ขอขอบคุณ คุณศิริศักดิ์ จังคศิริ ที่ช่วยให้คำชี้แนะ และให้ ความรู้ในทุกๆด้าน เกี่ยวกับการออกแบบ และการจัดทำโครงงาน

ขอขอบคุณ บริษัทไทยเทลเอนจิเนียริ่ง ที่ได้เอื้อเฟื้อสถานที่ในการจัดทำโครงงาน

ขอขอบคุณ อาจารย์ชินภัทร นันทจิวากรชัย ที่ให้ความรู้ และความเข้าใจ จนจบโครงงาน

เอกสารอ้างอิง

1.รศ.สมยศ จุณณะปิยะ, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS-51" ,คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ,301 หน้า,2541

2.รศ.ดร.กิตติ ไพฑูรย์วัฒนกิจ , "ไมโครโปรเซสเซอร์ และ ไมโครคอนโทรลเลอร์ " , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 321 หน้า,2538

3.ไกรวุฒิ โรจน์ประเสริฐสุด " เข้าใจ สร้าง เล่น ไมโครโปรเซสเซอร์ 2 " ,ซีเอ็ดยูเคชั่น,185 หน้า,2539